Homography Propagation and Optimization for Wide-Baseline Street Image Interpolation

Yongwei Nie, Zhensong Zhang, Hanqiu Sun, *Member, IEEE*, Tan Su, and Guiqing Li

Abstract—Wide-baseline street image interpolation is useful but very challenging. Existing approaches either rely on heavyweight 3D reconstruction or computationally intensive deep networks. We present a lightweight and efficient method which uses simple homography computing and refining operators to estimate piecewise smooth homographies between input views. To achieve the goal, we show how to combine homography fitting and homography propagation together based on reliable and unreliable superpixel discrimination. Such a combination, other than using homography fitting only, dramatically increases the accuracy and robustness of the estimated homographies. Then, we integrate the concepts of homography and mesh warping, and propose a novel homography-constrained warping formulation which enforces smoothness between neighboring homographies by utilizing the first-order continuity of the warped mesh. This further eliminates small artifacts of overlapping, stretching, etc. The proposed method is lightweight and flexible, allows wide-baseline interpolation. It improves the state of the art and demonstrates that homography computation suffices for interpolation. Experiments on city and rural datasets validate the efficiency and effectiveness of our method.

Index Terms—Image interpolation, street view synthesis, homography propagation, homography-constrained warping

1 Introduction

Interpolation between street views is an emerging problem in modern virtual street roaming applications such as Google StreetView and Bing StreetSide. The map engines maintain a large set of sparsely sampled street pictures, and provide operations to interactively explore among them. For example, a user can change from a street view to an adjacent but distant one by clicking on a directional arrow. To make the virtual exploration more immersive, the transition between two discrete street views should be as smooth as possible, which is a street view interpolation problem that demands interpolating intermediate views realistically.

However, there usually exists wide baseline spacing discrete street views. By "baseline", we mean the translation, rotation, or the combination of both that a camera undergoes to capture adjacent images. We follow the way of [1] to define various kinds of baselines. The median distance between the original KITTI images is 0.8 m [2], based on which (1) *natural baseline* is defined as 1.6 m, i.e., taking every other KITTI images as input for view interpolation, with the skipped images as ground truth for comparison; (2) *wide baseline* is 3.2 m by skipping three images; and (3) *very wide baseline* is 4.8 m by skipping five images.

Wide baseline brings large scalings, translations, and rotations of image patches. It also yields significant cropping and shearing of image boundaries, disappearance and reappearance of objects. In addition, a street view itself usually contains large textureless regions. All of these factors make it hard to interpolate between wide-baseline street images.

Traditional interpolation methods were usually designed for small-baseline (< 0.8 m) images. For example, the algorithms in [3], [4], [5], [6] were proposed to interpolate between video frames, the aim of which is to increase the frame rate of the video. Adjacent video frames are very similar such that pixel-level correspondence searching is possible. Most of the early optical flow methods evaluated on Middlebury benchmark [7] belong to the small-baseline category, as the objects in the Middlebury dataset usually undergo small displacements. Recent large displacement optical flow approaches [2], [8] can be regarded as medium-baseline algorithms. The development of these methods has been prospered with the release of KITTI [2] and MPI Sintel [9] benchmark. However, even in KITTI or Sintel dataset, the baseline between pairs of images is not wide enough which is only 0.8 m away.

For pairs of wide-baseline images (with baseline larger than 3.2 m), directly finding precise correspondences between them becomes very challenging. It is thus popular to seek support from additional information or constraints. For example, [10] interpolated between pairs of wide-baseline stereo images using epipolar constraints. The Google StreetView additionally acquired depth information by laser scanners when capturing street view images [11]. On a different research line, Image-based Rendering algorithms [12], [13] usually demanded a sufficient set of images that can be used to reconstruct a 3D scene. Recently, Flynn et al. [1] showed that deep networks can be used for wide-baseline view synthesis. Although existing wide-baseline algorithms can produce pleasant synthesis results, they rely on either advanced equipment, heavyweight 3D reconstruction or computationally intensive deep networks.

- *Y. Nie and G. Li are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China. E-mail: {nieyongwei, ligq}@scut.edu.cn.*
- *Z. Zhang, H. Sun, and T. Su are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong. E-mail: {zszhang, hanqiu, tsu}@cse.cuhk.edu.hk.*

Manuscript received 27 July 2016; revised 9 Oct. 2016; accepted 13 Oct. 2016. Date of publication 19 Oct. 2016; date of current version 1 Sept. 2017. Recommended for acceptance by S. Hu.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2016.2618878

In this paper, we propose a method that can directly interpolate between wide-baseline street views without additional constraints nor depth information. Our method divides a source image into superpixels, and tries to compute for each superpixel a parametric transformation model—i.e., homography—which can transform the superpixel to its target position in a corresponding target image. Commonly, the homographies of the superpixels are fitted from sparse feature correspondences. However, in wide-baseline street images, there exist many regions where homography fitting is not accurate enough as reliable feature correspondences are not sufficient in these regions due to occlusions, lack of textures, etc. To increase accuracy and robustness, we propose a homography estimation method that combines homography fitting and homography propagation together based on reliable and unreliable superpixel discrimination, where the reliable superpixels are suitable for homography fitting, while the unreliable superpixels are applied with homography propagation. The homography propagation scheme is actually a heuristic that associates each unreliable superpixel with a set of reliable superpixels and assigns the average homography of the reliable superpixels to the unreliable superpixel. The underlying assumption of this heuristic is that a unreliable superpixel and the corresponding set of reliable superpixels belong to the same plane, which is usually true in street images.

However, although the homography fitting and propagation strategy dramatically improves the quality of homographies, there still may exist small errors. More importantly, the strategy computes homography for each superpixel separately which cannot guarantee them to be piecewise smooth. These factors may lead to small artifacts of scaling, stretching, overlapping, and holes, etc. To further eliminate the small artifacts, we optimize the homographies by incorporating them into a homography-constrained mesh warping framework to enforce piecewise smoothness between neighboring homographies.

In summary, our main contribution is the proposed three-step homography computing and refining algorithm:

1) A robust reliable and unreliable superpixel discrimination algorithm based on spatial consistency of Approximate Nearest Neighbor Field (ANNF).
2) An accurate and robust homography estimation method that combines homography fitting and homography propagation together.
3) A homography-constrained warping formulation that enforces homography smoothness.

The rest of this paper is organized as follows. Section 2 introduces related work. Section 3 describes our approach in detail, including preprocessing (Section 3.1), reliable and unreliable superpixel discrimination (Section 3.2), combined homography fitting and propagation (Section 3.3), and homography optimization (Section 3.4). Section 4 gives experimental results. Conclusion is given in Section 5.

2 Related Work

Generally speaking, image interpolation is a high-level application of low-level dense correspondence techniques such as optical flow, motion analysis, etc. We mainly review the most relevant motion estimation algorithms, and roughly divide them into three categories: small-baseline, medium-baseline, and wide-baseline approaches.

Small-Baseline Interpolation. Early optical flow methods evaluated on Middlebury benchmark [7] are typical small-baseline algorithms. In the situation of small-baseline where objects undergo small displacements, pixel-to-pixel correspondence can be found in a global and optimal way by solving the variational model proposed by Horn and Schunck in [14]. Many methods has been developed to improve the original variational model to cope with illumination variations, blurring, discontinuities, and occlusions, etc. Please refer to [7] for a more detailed survey. Generally, small-baseline optical flow methods cannot handle pairs of wide-baseline images, as searching pixel-to-pixel correspondence is challenging when two images are very different.

A typical application of small-baseline methods is Motion-Compensated Frame Interpolation which interpolates between video frames to increase the frame rate of a video. Since sequential video frames are very similar, many domain-specific block-based or pixel-based motion estimation algorithms have been investigated. Block-based methods [3], [4] divided a frame into non-overlapping blocks, and then for each block searched for the most similar block in the following frame. On pixel level, Mahajan et al. [5] observed that a pixel in the interpolated frames traced out a path in the source image, based on which they moved pixels gradients along the path, and finally recovered intermediate frames by Poisson reconstruction. Stich et al. [6], [15] found edges and homogeneous regions in images for matching, yielding a dense motion field between images. However, block-based and pixel-based approaches usually demand subtle low-level operations which are not effective for large-scale scene interpolation.

Medium-Baseline Interpolation. Recently, optical flow research community has greatly pushed forward the development of large displacement optical flow estimation. Large displacement usually comes from two sources: fast motion of an object (e.g., MPI Sintel benchmark [9]), and large translation or rotation of a camera (see KITTI benchmark [2]). To recover large displacement optical flow, traditional methods [16], [17] incorporated the variational optical flow model [14] into a coarse-to-fine multi-scale scheme, which however cannot handle fast motions. In [18], [19], the variational model was initialized by sparse feature correspondences or approximate nearest neighbor field which help the variational model to escape from local minima. Modern methods [20], [21], [22] improved [18], [19] by proposing more sophisticated feature matching algorithms that are specially tailored to the optical flow problem. For example, DeepFlow [20] presented a descriptor matching algorithm, and EpicFlow [21] found dense matching by edge-preserving interpolation from a sparse set of matches. From different angles, Menze et al. [8] first found discrete pixel-accurate optical flow, then refined it to sub-pixel level. Yang et al. [23] optimized homogeneous regions and their homographies simultaneously, which however would fail in large textureless regions. Bao et al. [24] obtained large displacement optical flow by increasing the smoothness of PatchMatch [25]. In this paper, we compare our method with DiscreteFlow [8] which is a representative large displacement optical flow method.

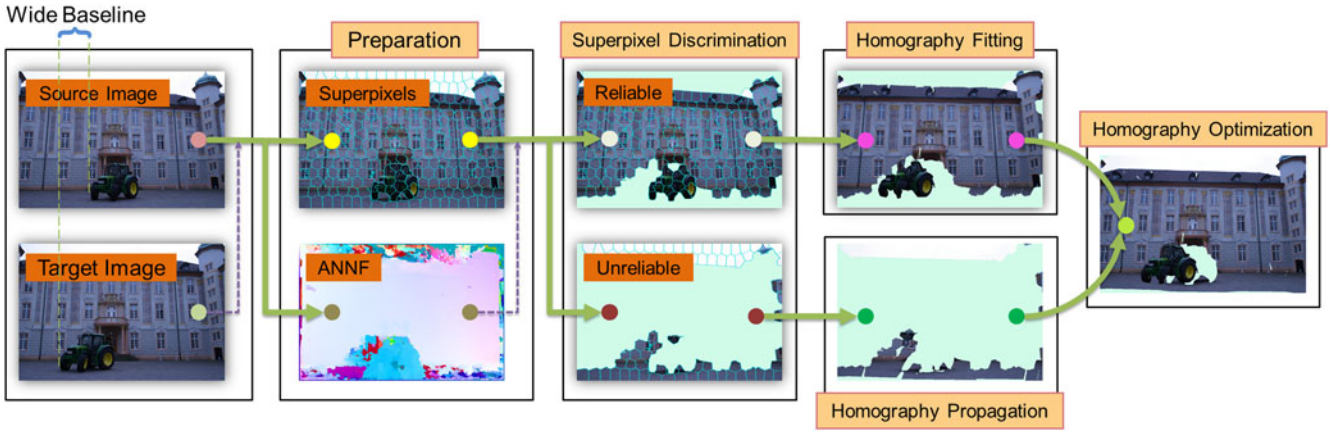


Fig. 1. Overview of our algorithm that computes for a source image piecewise smooth homographies with respect to a target image. Given a pair of images with wide baseline, we first segment the source image into superpixels and compute the Approximate Nearest Neighbor Field (ANNF) from the source image to the target image in the preparation step. We then discriminate reliable superpixels from unreliable superpixels by the consistency of ANNF. Next, we compute homographies for reliable superpixels by homography fitting, and compute homographies for unreliable superpixels by homography propagation. Finally, the estimated homographies undergo an optimization to enforce smoothness between them.

Wide-Baseline View Synthesis. Wide-baseline view synthesis can be achieved if assuming enough additional information or constraints. For example, Google StreetView [11] required depth information from laser scanners. Wide-baseline stereo methods [10], [26], [27] relied on binocular images. Recent methods of image-based rendering [12], [13], [28] reconstructed 3D model of a scene, and compensated for the errors of the reconstruction by silhouette-aware warping [12] or depth synthesis [13], etc. However, they demand a collection of images of the scene as input, and are not robust enough for rural scenes (such as those of KITTI [2]) where 3D reconstruction may fail. Recently, some researchers [1], [29], [30] tried to apply learning methods to novel view synthesis. For example, Flynn et al. [1] developed a DeepStereo system that trained a deep network to predict novel views from the world's imagery. The learning-based method required a large amount of training data and much training time. Camera poses were also required when training. By contrast, our method only needs two images as input, which is more lightweight and flexible.

3 Approach

Our aim is to smoothly transform from a source street view I_s to a corresponding target view I_t. Our method divides I_s into superpixels, and views each superpixel as a small plane which corresponds to somewhere in the target image I_t by a homography. We try to compute the homographies of superpixels as accurate and robust as possible. As shown in Fig. 1, we propose a three-step homography computing and refining algorithm: (1) reliable and unreliable superpixel discrimination: a superpixel is identified as either reliable or unreliable; (2) homography fitting and propagating: the homographies of reliable superpixels are computed by homography fitting while the homographies of unreliable superpixels are computed by homography propagation; (3) homography optimization: the homographies are further optimized to enforce smoothness between adjacent homographies. We compute both the forward homographies from I_s and I_t and the backward homographies from I_t to I_s. With the estimated homographies, intermediate images can be interpolated and blended.

3.1 Preparation

Before introducing our three-step algorithm, we do some preparations in this section, as shown in the second column of Fig. 1. We oversegment the source image I_s into a collection of superpixels. We also compute the Approximate Nearest Neighbor Field from I_s to I_t which helps to discriminate reliable superpixels from unreliable ones.

We employ the method of [31] for superpixel segmentation. Superpixel serve as the building block of the proposed algorithm. Since a superpixel usually has homogeneous color, we can view each superpixel as a small plane and compute a homography for it. In this paper, we set the superpixel size at 500 pixels. For a 800×600 image, there are about 1,000 superpixels. We denote the set of superpixels of an image by $\mathcal{S} = \{S_i | i \in \{0, \ldots, n-1\}\}$.

We use the generalized PatchMatch method [32] to compute the ANNF from I_s to I_t which is a dense correspondence field indicating for each patch of I_s the most similar patch in I_t. A source patch can be translated, scaled, and rotated to find the best match in the target image. The image at the bottom of the second column of Fig. 1 illustrates the computed ANNF.

3.2 Reliable and Unreliable Superpixel Discrimination

Given the superpixels and ANNF computed in the last section, we identify whether a superpixel is reliable or not for homography fitting by measuring the consistency of the ANNF of the superpixel. The ANNF indicates pixel correspondences between the source and target images. If the correspondences of the pixels of the whole superpixel are consistent enough with each other, we view the superpixel as reliable.

We employ and improve the method of [33] to measure the reliability of a superpixel. First, we define two pixels as consistent if their ANNF correspondence vectors are similar. Specifically, let P and Q be two patches of image I_s, and P' and Q' be the corresponding patches of image I_t by ANNF, with p, q, p', and q' be the centers of the four patches and $p' = p + \mathcal{F}(p)$, $q' = q + \mathcal{F}(q)$, where $\mathcal{F}$ indicates the ANNF. Normally, to compute the consistency error $C(p, q)$ between

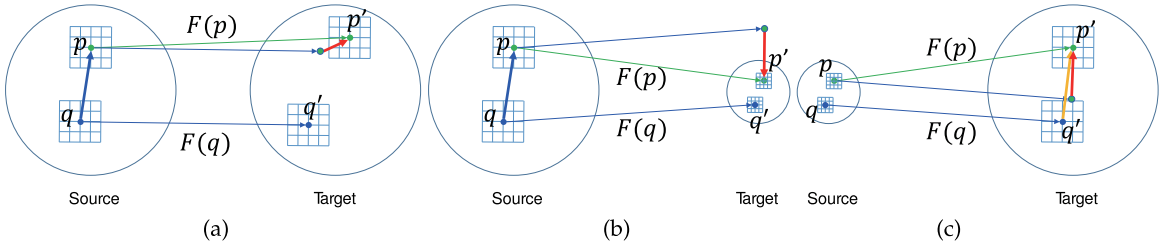


Fig. 2. Three situations when computing superpixel consistency value: (a) No scale. (b) Scale down. (c) Scale up. Red arrow denotes $\mathcal{F}(p) - \mathcal{F}(q)$, blue arrow denotes $p - q$, and yellow arrow denotes $p' - q'$.

pixels p and q, we just need to compare the difference between $\mathcal{F}(p)$ and $\mathcal{F}(q)$, i.e., $C(p,q) = ||\mathcal{F}(p) - \mathcal{F}(q)||_2$, as shown in Fig. 2a. However, since the generalized PatchMatch algorithm contains scaling transformation, simple $||\mathcal{F}(p) - \mathcal{F}(q)||_2$ is not adequate in the examples of Figs. 2b and 2c. In Fig. 2b, the patches are scaled down. Although it is actually consistent between p and q, $||\mathcal{F}(p) - \mathcal{F}(q)||_2$ is very large. In this situation, we need to normalize $||\mathcal{F}(p) - \mathcal{F}(q)||_2$ by the distance between p and q. In Fig. 2c, the patches are scaled up. It is another case where $||\mathcal{F}(p) - \mathcal{F}(q)||_2$ is large but p and q is consistent. In this case, we normalize $||\mathcal{F}(p) - \mathcal{F}(q)||_2$ by the distance between p' and q'. Finally, the consistency error between two pixels p and q is defined as

$$C(p,q) = \begin{cases} \frac{||\mathcal{F}(p)-\mathcal{F}(q)||_2}{||p-q||_2}, & \text{if } s(P) \leq 1 \\ \frac{||\mathcal{F}(p)-\mathcal{F}(q)||_2}{||p'-q'||_2}, & \text{if } s(P) > 1, \end{cases} \quad (1)$$

where, $s(P)$ is the scaling factor of patch P computed by the generalized PatchMatch algorithm [32]. We then define a delta function

$$\delta(p,q) = \begin{cases} 1, & \text{if } C(p,q) < \delta_1 \\ 0, & \text{otherwise,} \end{cases} \quad (2)$$

where δ_1 is a threshold value to determine whether a pair of pixels is consistent or not.

To compute the reliability of a whole superpixel S_i, we randomly sample (in order to save time) $K = \sqrt{|S_i|}$ pairs of pixels in the superpixel, where $|S_i|$ is the number of pixels of S_i. The consistency value $C(S_i)$ of a superpixel S_i is

$$C(S_i) = \frac{\sum_{i=1}^{K} \delta(p_i, q_i)}{K}. \quad (3)$$

Finally, we use a threshold δ_2 to select the set of superpixels that have higher consistency values, which constitute the set of reliable superpixels $\mathcal{R}$ we seek

$$\mathcal{R} = \{S_i | \forall i \in \{0, \ldots, n-1\}, C(S_i) > \delta_2\}, \quad (4)$$

with the remaining superpixels as unreliable: $\mathcal{U} = \mathcal{S} - \mathcal{R}$.

In our experiments, we set δ_1 and δ_2 as 2 and 0.9 (please refer to Section 4 for more discussion). Figs. 3a and 3b show the identified reliable superpixels $\mathcal{R}$ and unreliable superpixels $\mathcal{U}$, respectively.

3.3 Homography Estimation Combining Fitting and Propagation

3.3.1 Homography Fitting

For every reliable superpixel $S_i \in \mathcal{R}$, we directly fit a homography from the superpixel's ANNF. Each pixel of S_i has a corresponding pixel in the target image via ANNF. Most of the correspondences are reliable except a few outliers. We fit an accurate homography to those correspondences by the RANSAC algorithm [34] that is robust to outliers.

Fig. 3c shows the result of transforming the reliable superpixels of $\mathcal{R}$ by their homographies. The transformation is consistent and has no error. For comparison, we also fit homographies for the unreliable superpixels of $\mathcal{U}$. The transformed result of $\mathcal{U}$ is shown in Fig. 3d. It can be seen that the transformation is not consistent and contains many wrong mappings. The comparison shows that homography fitting on unreliable superpixels are problematic, and demonstrates the effectiveness of the reliable and unreliable superpixel discrimination algorithm.

3.3.2 Homography Propagation

Homography fitting is problematic in unreliable regions. Possible reasons include occlusion, boundary cropping and shearing due to large camera translation, lack of texture, etc., which lead to errors in patch matching and reduce the spatial consistency of ANNF. To estimate homographies for unreliable superpixels, we use a homography propagation heuristic that propagates the reliable homographies of $\mathcal{R}$ to the unreliable superpixels $\mathcal{U}$. Specifically, for each unreliable superpixel S_i, we associate it with a collection of reliable superpixels $\mathcal{S}_C$, and then assign the homography computed from $\mathcal{S}_C$ to S_i. To make the propagation valid, it should be ensured as much as

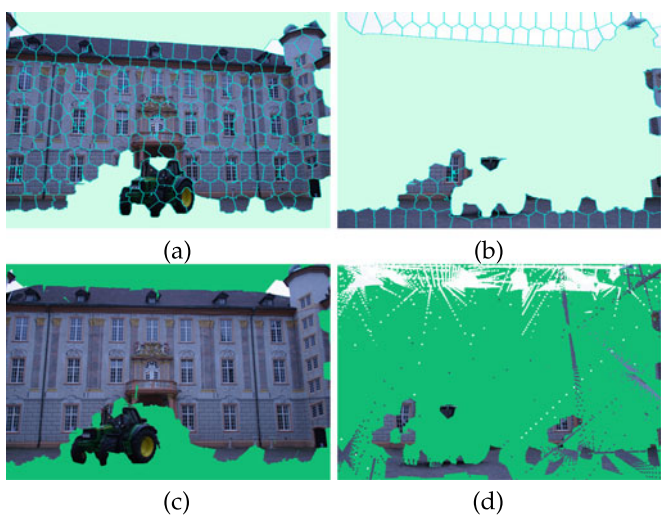


Fig. 3. (a) Reliable superpixels identified by the discrimination algorithm. (b) The remaining unreliable superpixels. (c) Image obtained by transforming the reliable superpixels according to their homographies estimated by homography fitting. (d) Image obtained by transforming the unreliable superpixels according to their homographies estimated by homography fitting.

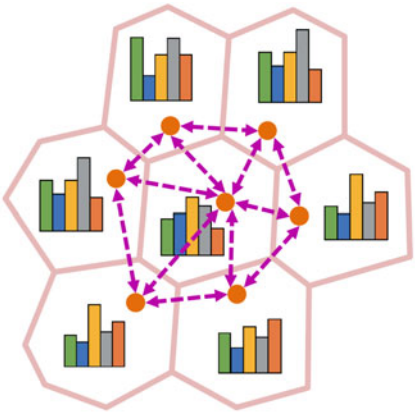


Fig. 4. Superpixel graph. Every superpixel is considered as a node in the graph, and edges are added to the graph when two superpixels share common boundaries, the weights of the edges are the χ^2 distances between RGB histograms of the associated superpixels.

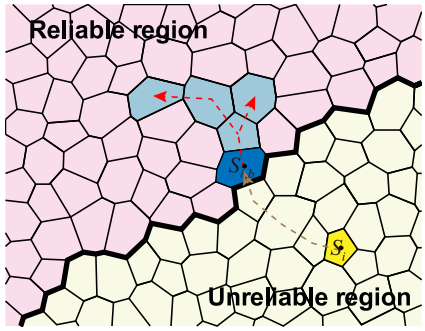


Fig. 5. Illustration of homography propagation. The reliable superpixels and unreliable superpixels are separated by a bold black boundary. For each unreliable superpixel S_i (the yellow one), we associate a set of reliable superpixels $\mathcal{S}_C$ to it. To this end, we first find $S_{i,b}$ (the blue superpixel) on the boundary of $\mathcal{R}$ that is nearest to S_i according to $D(S_i, S_{i,b})$, and then locate any S_j (the light blue ones) if $D(S_j, S_{i,b}) < \gamma$ and the corresponding path length between S_j and $S_{i,b}$ is less than 3.

possible that S_i and $\mathcal{S}_C$ actually belong to the same plane, which demands those superpixels are close to each other in both color space and spatial space, or even belong to the same object. Based on those thoughts, we propose a propagation heuristic that works as follows. We construct a superpixel graph on which a distance metric is defined that can depict both color similarity and spatial proximity between superpixels. The distance metric is then used to associate similar superpixels for homography propagation. When necessary, we segment the source image into object layers, and let the propagation occur only within the same layer.

Homography Propagation on Superpixel Graph. As shown in Fig. 4, the superpixel graph is constructed by viewing each superpixel of the source image as a node of the graph, and edges are added to the graph when two superpixels share a common boundary. We define the weight of each edge as the color similarity between the associated superpixels. To define the color similarity, we follow the way of [35] to create normalized color histogram for each superpixel. The histogram of every color channel has 16 bins, and three histograms of RGB are concatenated as a 48 D descriptor $\mathcal{H}_{RGB}[S_i]$. The color similarity between two superpixels is computed as the χ^2 distance between the two histograms.

Given the superpixel graph, for any pair of superpixels S_i and S_j, there exist a lot of paths $\mathbb{P}[S_i \rightarrow S_j]$ from S_i to S_j through the graph. Let ρ be one such path of length $|\rho|$, we define the energy of the path using the same way of [13]

$$d(S_i \xrightarrow{\rho} S_j) = \sum_{t=0}^{|\rho|-1} \chi^2(\mathcal{H}_{RGB}[\rho(t)], \mathcal{H}_{RGB}[\rho(t+1)]), \quad (5)$$

with $\rho(0) = S_i$ and $\rho(|\rho|) = S_j$. By finding the path $\hat{\rho}$ that has the minimal energy

$$\hat{\rho} = \underset{\rho \in \mathbb{P}[S_i \rightarrow S_j]}{\arg\min}\ d(S_i \xrightarrow{\rho} S_j), \quad (6)$$

we define the distance between S_i and S_j be $D(S_i, S_j) = d(S_i \xrightarrow{\hat{\rho}} S_j)$ which can depict both color similarity and spatial proximity between S_i and S_j. The minimal energy paths can be easily computed by the Dijkstra shortest path algorithm [13].

Recall that we need to search a set of reliable superpixels $\mathcal{S}_C$ for $S_i \in \mathcal{U}$. One way is to compute the distance between S_i and every reliable superpixel of $\mathcal{R}$, and find the set of reliable superpixels with minimal distances, as done in [13]. However, such heuristic would produce $\mathcal{S}_C$ that are scattered over the source image, suffering from great risk of coming from different planes.

We need $\mathcal{S}_C$ themselves to be compact in the spatial space and thus use a different heuristic. As shown in Fig. 5, for an unreliable superpixel S_i (the yellow one), we first find the blue superpixel $S_{i,b}$ on the boundary of $\mathcal{R}$ which is nearest to S_i according to $D(S_i, S_{i,b})$, and push $S_{i,b}$ into $\mathcal{S}_C$. Then, for any reliable superpixel S_j, if the shortest path between $S_{i,b}$ and S_j passes no more than three superpixels, and $D(S_{i,b}, S_j) < \gamma$ (in this paper, we set γ as 1.5, for more discussion please refer to Section 4), we push S_j into S_C. In this way, we find the set of superpixels $\mathcal{S}_C$ for S_i, i.e., the blue and light blue ones in Fig. 5. Once we find $\mathcal{S}_C$, all the ANNF correspondences of $\mathcal{S}_C$ are used to fit a homography that is finally assigned to the unreliable superpixel S_i.

Propagation Confined in Object Layers. The propagation described above is built on the assumption that superpixels similar in color and close in distance are on the same plane. However, the assumption is a little weak. There are still chances of propagating between different planes. In this section, we make a stronger assumption that superpixels of the same object are on the same plane, which is mostly true in street view images, such as wall of building, road, etc. In this sense, we automatically segment the source image into object layers, and allow the homography to propagate only in the same layer to improve the robustness of our method.

We adopt the hierarchical graph segmentation method [35], [36] to segment the source image into initial object layers. The method proceeds in a bottom-up fashion. The image is initially over-segmented into small superpixels. Then, two small and adjacent superpixels are merged into a bigger region if they are similar enough in color space. The merging scheme continues iteratively to obtain larger objects until a pre-defined granularity of object size achieved. Fig. 6a shows the layer segmentation result of [35]. However, there are many small fragments. We improve it by two heuristics. First, if more than 50 percent of the boundary of a segment is shared by its neighboring

(a) (b)

Fig. 6. (a) Hierarchical segmentation result of [35]. (b) Our object layers after improving (a).

segment, the two segments are merged. Second, if a segment is too small, it is merged with its biggest neighboring segment. Fig. 6b shows our layer segmentation result after improvement.

Once object layer segmentation on the source image is finished, we propagate homographies from reliable superpixels to unreliable superpixels of the same layer. Please be noted that layer boundaries may not coincide with the boundaries of our superpixels for homography fitting and propagation. To enforce the consistency, we divide the superpixels if a layer boundary cross them.

Fig. 7 shows the comparison between homography fitting and homography propagation on unreliable superpixels, from which we can see that our homography propagation scheme dramatically improves the accuracy of the homographies of unreliable superpixels.

3.4 Homography Optimization by Mesh Warping

The homographies computed above may still contain errors due to two issues. First, since the propagation is performed for each superpixel individually, different unreliable superpixels may be associated with different sets of reliable superpixels. Separately computed homographies in this way are not guaranteed to be piecewise smooth. Second, the accuracy of homography propagation will decrease with the increase of the distance of the propagation. The two issues may lead to artifacts of small holes, overlaps, scales, and stretches in the interpolated images, as shown in Fig. 8a.

We propose to solve the above two problems by an optimization which further regularises the homographies to make them piecewise smooth. We achieve our goal by integrating the homographies into the as-similar-as possible image warping framework [37], [38], and propose a novel homography-constrained warping formulation. During the optimization, homographies play the role of constraints to make the warping not deviate from the original transformation (homographies) too much, while the first-order

(a) (b)

Fig. 7. Comparison between homography fitting and homography propagation on unreliable superpixels. (a) The result of homography fitting. (b) The result of homography propagation.

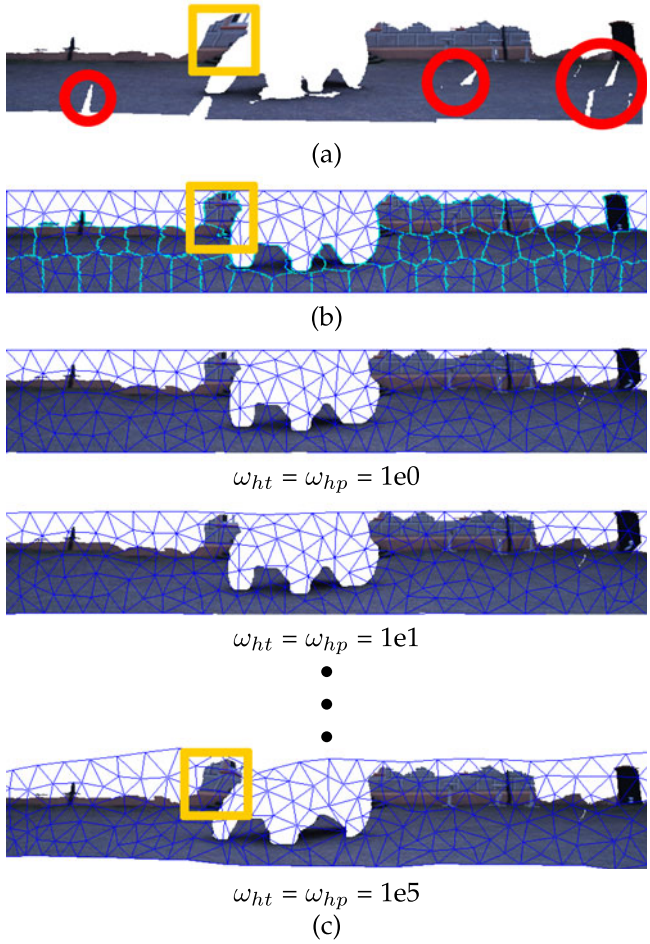


Fig. 8. (a) The transformed image by the homographies before optimization, which contains artifacts: The region in the square is stretched too much, and the regions in the circles show hole and overlap artifacts. (b) Source image overlaid with mesh. (c) Result after the homography optimization scheme under different sets of weights.

continuity of the warped mesh makes the optimized homographies piecewise smooth.

Specifically, we overlay a Delaunay triangle mesh on the source image (or on each layer if the source image is segmented into several object layers as shown in Fig. 8b). Let $H_{S_i} = [h_{i1}, h_{i2}, h_{i3}; h_{i4}, h_{i5}, h_{i6}; h_{i7}, h_{i8}, 1]$ be the optimized homography of superpixel S_i, and $\mathbf{H} = \{H_{S_i}\}$ be the set of all optimized homographies. Denoting by $\mathbf{V}$ the deformed mesh vertices. Our new warping formulation optimizes $\mathbf{V}$ and $\mathbf{H}$ simultaneously

$$E(\mathbf{V}, \mathbf{H}) = \omega_{data} E_{data}(\mathbf{V}) + \omega_{sim} E_{sim}(\mathbf{V}) + \omega_{ht} E_{ht}(\mathbf{V}, \mathbf{H}) + \omega_{hp} E_{hp}(\mathbf{H}), \tag{7}$$

where E_{data} is data term, E_{sim} is similarity term, E_{ht} is homography transformation term, and E_{hp} is homography persistence term. The data term E_{data} and similarity term E_{sim} are borrowed from [37], which are only related to the deformed mesh vertices $\mathbf{V}$. The homography transformation term E_{ht} and homography persistence term E_{hp} are two newly introduced terms. The homography transformation term is used to enforce the consistency between the optimized mesh and homographies, and the homography persistence term is used to ensure that that the optimized homographies do not deviate much from their

corresponding original homographies. ω_{data}, ω_{sim}, ω_{ht} and ω_{hp} are weights of the four terms.

Data Term. As stated in [37], the data term is used to make a set of user indicated control points approach their target positions as much as possible during the warping. In our situation, the pixel-to-pixel correspondences in reliable regions are used as control points. Instead of using all the reliable pixels, we just evenly sample a set of points over reliable superpixels as control points.

Let $\hat{p}_k$ be the kth sampled control point in the source image, and p_k be its correspondence in the target image. Typically, the control points do not coincide with mesh vertices. Suppose that $\hat{p}_k$ is enclosed in triangle $\hat{\triangle}_k = [\hat{v}_{k1}, \hat{v}_{k2}, \hat{v}_{k3}]^T$ of source mesh $\hat{\mathbf{V}}$, we represent $\hat{p}_k$ by the mean value coordinates $w_k = [w_{k1}, w_{k2}, w_{k3}]^T$ with respect to the three vertices of the triangle, i.e., $\hat{p}_k = w_k^T \hat{\triangle}_k$ [39]. The warped position of $\hat{p}_k$ is $w_k^T \triangle_k$, where $\triangle_k = [v_{k1}, v_{k2}, v_{k3}]^T$ is the new positions of triangle vertices. Then, the data term is

$$E_{data}(\mathbf{V}) = \sum_k \|w_k^T \triangle_k - p_k\|^2. \tag{8}$$

Similarity Term. The similarity term ensures that the transformed triangles are as similar as possible to the corresponding original triangles. We use the same similarity term as in [37], which penalizes the deviation of each output triangle from a similarity transformation of the corresponding input triangle. Specially, let $(\hat{v}_{k1}, \hat{v}_{k2}, \hat{v}_{k3})$ and (v_{k1}, v_{k2}, v_{k3}) be the vertices of the kth triangle in the input mesh and the output mesh, respectively. Every vertex in a triangle can be locally represented by the resting two vertices as

$$\hat{v}_{k1} = \hat{v}_{k2} + a(\hat{v}_{k3} - \hat{v}_{k2}) + bR_{90}(\hat{v}_{k3} - \hat{v}_{k2}), \tag{9}$$

where $R_{90} = [0, 1; -1, 0]$, a and b are the relative coordinates of $\hat{v}_{k1}$ locally defined by $\hat{v}_{k2}$ and $\hat{v}_{k3}$. The expected position of v_{k1} in the output mesh is thus

$$v_{k1}^{expected} = v_{k2} + a(v_{k3} - v_{k2}) + bR_{90}(v_{k3} - v_{k2}). \tag{10}$$

Then the deviation error for vertex v_{k1} is represented as $\|v_{k1} - v_{k1}^{expected}\|^2$, and the similarity term is defined as

$$E_{sim}(\mathbf{V}) = \sum_k \sum_{i=1,2,3} \|v_{ki} - v_{ki}^{expected}\|^2. \tag{11}$$

Homography Transformation Term. The homography transformation term enforces the consistency between the optimized mesh and homographies. Given an optimized homography H_{S_i}, and a source vertex $\hat{v}_k$ which belongs to S_i, the source vertex will be projected to $H_{S_i} \cdot \hat{v}_k$ by H_{S_i}. From another perspective, the new position of $\hat{v}_k$ is v_k of the warped mesh. The homography transformation term ensures $H_{S_i} \cdot \hat{v}_k$ equals to v_k

$$E_{ht}(\mathbf{V}, \mathbf{H}) = \sum_{k,i} \left\|v_k - H_{S_i} \cdot \hat{v}_k\right\|^2. \tag{12}$$

Homography Persistence Term. The homography persistence term ensures that the new optimized homographies H_{S_i} resemble their corresponding original homographies $\hat{H}_{S_i}$ as much as possible. For every superpixel, we sample $N = 10$ pixels $(\hat{x}_k, \hat{y}_k)$, and define the homography persistence term as

$$E_{hp}(\mathbf{H}) = \sum_i \sum_{k=1}^{N} \left\|H_{S_i} \cdot \hat{v}_k - \hat{H}_{S_i} \cdot \hat{v}_k\right\|^2. \tag{13}$$

Energy Minimization. We minimize Eq. (7) using Gauss-Newton Method with a backline search strategy [40]. The parameters ω_{data}, ω_{sim}, ω_{ht} and ω_{hp} can balance the weights of the energy terms. To demonstrate the usage of the homography terms, we fixed the first two parameters as 1, and took six values, from $1e^0$ to $1e^5$ by a factor of 10, for the last two parameters. Fig. 8c shows our results after warping-based homography optimization, under different sets of weights. The artifacts of gaps, holes, and overlaps between the transformed superpixels are removed after warping. With the increase of ω_{ht} and ω_{hp}, the shape of the squared region is preserved without stretching, and at the same time the squared region is moved to its desired position. To produce the results in this paper, we experimentally set $\omega_{data} = 1$, $\omega_{sim} = 1$, $\omega_{ht} = 1e5$ and $\omega_{hp} = 1e5$, respectively.

Please be noted that our homography-constrained warping plays the same role as the local warping of [13], both of which serve as a post processing operator. However, unlike the local warping that needs to be performed per frame to compute the warped image, our homography-based warping just need to be executed only once. The next section uses the optimized homographies to interpolate intermediate images.

3.5 Homography-Based Interpolation

Based on the estimated homographies, we use a simple method that can smoothly interpolate any intermediate image I^i at time $i \in (0, 1)$ between a source street image I_s and a target street image I_t. Our method is composed of three steps (see Fig. 9):

Step 1: Motion computing. First, given the estimated homographies, we can compute a forward motion field $\mathbf{u}_0$ from I_s to I_t, and also a backward motion field $\mathbf{u}_1$ from I_t to I_s inversely.

Let S_i be a superpixel of I_s, and H_{S_i} be the estimated homography of S_i. Every pixel $\mathbf{x}$ of S_i in I_s is related to its target position $\mathbf{x}'$ in I_t by H_{S_i}, i.e., $\mathbf{x}' = H_{S_i} \cdot \mathbf{x}$. Hence the motion of $\mathbf{x}$ from I_s to I_t is $H_{S_i} \cdot \mathbf{x} - \mathbf{x}$. In this way, we can compute the motion flow $\mathbf{u}_0$ from I_s to I_t by $\mathbf{u}_0(\mathbf{x}) = H_{S_i} \cdot \mathbf{x} - \mathbf{x}$.

$\mathbf{u}_1$ is computed in the same way, but now I_t is the source image, and I_s is the target image.

Step 2: Intermediate image rendering. To interpolate an intermediate image I^i at time i, we first warp I_s and I_t to time i according to their motion field $\mathbf{u}_0$ and $\mathbf{u}_1$, respectively. For any pixel $\mathbf{x}$ of I_s, its new position in the warped image I_s^i is $\mathbf{x} + i \cdot \mathbf{u}_0(\mathbf{x})$. Then, we can render I_s^i by $I_s^i(round(\mathbf{x} + i \cdot \mathbf{u}_0(\mathbf{x}))) = I_s(\mathbf{x})$.

In the same way, we can render I_t to time i by $I_t^i(round(\mathbf{x} + (1 - i) \cdot \mathbf{u}_1(\mathbf{x}))) = I_t(\mathbf{x})$.

Step 3: Multiband blending. From Fig. 9, we can see that I_s^i is complementary with I_t^i. The two intermediate images I_s^i and I_t^i are then blended together by pyramid-based multiband blending [41] to produce the final result I^i, i.e., $I^i = multi\_blending(I_s^i, I_t^i)$.

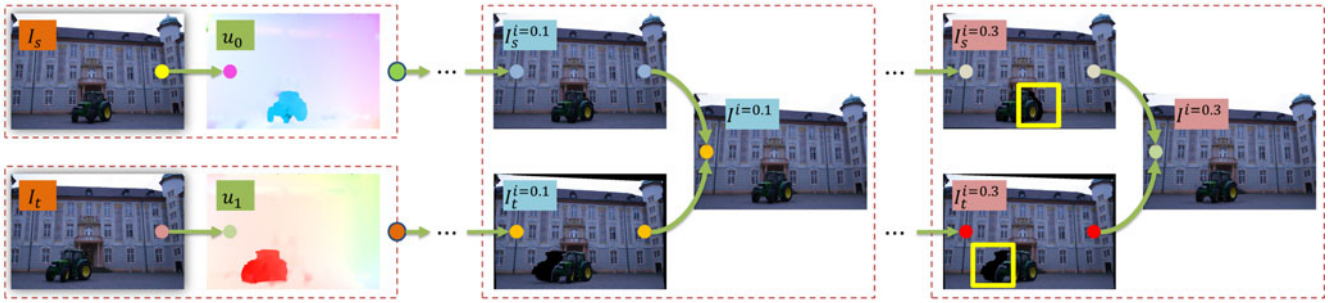


Fig. 9. Homography-based bidirectional interpolation. First, given the estimated homographies, we can compute a forward motion field $\mathbf{u}_0$ from I_s to I_t, and also a backward motion field $\mathbf{u}_1$ from I_t to I_s inversely. We then render the intermediate images I_s^i and I_t^i. Finally, the two intermediate images I_s^i and I_t^i are blended together by pyramid-based multi-band blending [41] to produce the final result I^i by $I^i = multi\_blending(I_s^i, I_t^i)$.

4 Experiments

We do experiments on various kinds of street images, collected from Google StreetView and KITTI datasets, varied from city to rural scenes. We compare our method with Google StreetView [11], DiscreteFlow [8], Image-based Rendering (IBR) methods [12] and [13], and DeepStereo [1]. The comparisons show that our method is much lightweight than previous methods, but can match or even surpass them in result quality.

4.1 Performance

We have developed our street view interpolation algorithm in C++ and run it on an Intel Xeon E5-1620 v2 3.7 GHz CPU computer with 8 GB memory. Table 1 shows the time consumed by our method on different examples. Different examples have different resolutions, and also different number of superpixels. We can see from Table 1 that larger images or more superpixels may lead to more computation time, and the most time-consuming stages in the proposed algorithm are the stages of homography propagation and homography optimization. This is because the homography propagation stage contains many RANSAC operations while the homography optimization stage needs to solve a large linear system. Overall, our method is lightweight, spending no more than 2 minutes on all the experiments of this paper.

We did a time comparison between DiscreteFlow [8] and our method. Both of the two methods take two images as input and compute correspondences between the two images. We used the code provided by the authors of DiscreteFlow, and run it on the same computer as our method. The last column of Table 1 shows the time consumed by DiscreteFlow. When there are more superpixels, the performance of our method matches that of DiscreteFlow. But when there are less superpixels, our method is faster than DiscreteFlow.

Both DiscreteFlow and our method are much more efficient than IBR methods [12] and [13] and DeepStereo [1] (we do not discuss about Google StreetView as the algorithm is not public). As stated in [12] and [13], the IBR methods "silhouette" and "depth synthesis" spent 30 minutes to 2 hours to prepare for view synthesis, while for DeepStereo it usually takes about hundreds of hours to train its deep network on high-performance computers. By contrast, since our algorithm depends only on simple homography computing and refining operators, it consumes time no more than 2 minutes, which is very lightweight. Besides, our algorithm can be perfectly parallelled to process a sequence of images. Since it is lightweight, our method can be more flexibly deployed on consumer device.

As for the rendering performance, Google StreetView [11], DiscreteFlow [8], IBR methods [12] and [13], and our method can all render an intermediate view in real time. DiscreteFlow and our method used the same rendering algorithm of Section 3.5, and can achieve about 45 FPS for 800×600 images with 500 superpixels. As reported in [12] and [13], the two state-of-the-art IBR methods obtained on average about 25 FPS on a laptop with similar computing capacity as our computer. By contrast, DeepStereo is much slower which used about 12 minutes to render just one 512×512 image on a multi-core workstation, which can only be used offline.

TABLE 1
Time Comparison with DiscreteFlow [8]

Example	Res	Sp Num	Sp Seg	PM	Sp Dis	Fit	Prop	Opt	Total	DiscreteFlow
First two columns of Fig. 12	1,450×779	500	4.71 s	6.02 s	0.05 s	4.93 s	59.06 s	23.28 s	98.04 s	95.96 s
Last two columns of Fig. 12	1,414×778	500	4.67 s	5.53 s	0.04 s	6.18 s	58.17 s	5.41 s	79.99 s	88.18 s
Third column of Fig. 13	1,242×375	500	2.03 s	2.42 s	0.04 s	3.34 s	23.27 s	1.77 s	32.86 s	38.75 s
Middle of Fig. 14	800×533	300	1.97 s	1.53 s	0.06 s	1.76 s	3.30 s	0.86 s	9.48 s	39.18 s
Left of Fig. 14	800×533	1,000	1.97 s	2.00 s	0.05 s	1.92 s	30.00 s	2.18 s	38.12 s	40.94 s
Right of Fig. 14	800×533	1,000	1.89 s	1.55 s	0.05 s	1.94 s	36.77 s	2.27 s	44.47 s	39.59 s
Last two columns of Fig. 15	800×533	500	1.86 s	1.94 s	0.07 s	3.23 s	2.38 s	2.06 s	11.54 s	37.90 s
Third column of Fig. 10	800×600	500	2.34 s	2.37 s	0.06 s	1.50 s	3.32 s	1.92 s	11.51 s	42.37 s

*"**Res**" represents resolution of image "**Sp Num**" is number of superpixels, "**Sp Seg**" gives time used in superpixel segmentation. One by one, "**PM**", "**Sp Dis**", "**Fit**", "**Prop**", and "**Opt**" mean the time used in the stages of ANNF computation, reliable and unreliable superpixel discrimination, homography fitting, homography propagation, and homography optimization. "**Total**" and "**DiscreteFlow**" present the total time used by our method and DiscreteFlow.*

Fig. 10. Row 1 shows results of homography fitting. Row 2 shows results of homography fitting and propagation. Row 3 shows results of homography fitting, propagation, and optimization. Holes are marked. Please refer to supplemental video for scale and stretch artifacts.

4.2 Stage Influence and Parameter Setting

The proposed algorithm includes three steps. In this section, we discuss the importance of each step and how different parameter settings would influence the final results. The first stage, i.e., the stage of reliable and unreliable superpixel discrimination, is the basis of the second stage. Without the first stage, we do not know whether a superpixel is reliable or not. In this situation, our algorithm would be degenerated to compute homographies all by homography fitting, which performs badly in wide-baseline interpolation as shown in the first row of Fig. 10. Based on the first stage, the second stage can then combine homography fitting and propagation together. With propagation, the accuracy and robustness of the estimated homographies are dramatically improved. Most of the errors shown in the first row of Fig. 10 are removed in the second row. The third stage, i.e., the stage of homography optimization, further removes small artifacts of the first two stages, making the final results visually better (see the third row of Fig. 10). In total, the first two stages are the most important components to make our wide-baseline street view interpolation workable, while the third stage plays the role of post-processing which is indispensable.

In the first stage, we have two parameters δ_1 and δ_2 to control whether a superpixel is reliable or not, where δ_1 in Eq. (2) is a threshold that checks the consistency of a pair of pixels, and δ_2 in Eq. (4) is a threshold that checks the reliability of a superpixel. A larger δ_1 (or a smaller δ_2) makes a superpixel pass the reliability check easier, which means there will be more superpixels classified as reliable. Fig. 11a shows such tendency, where we did experiments on the dataset of the first column of Fig. 10 and took 20 values for

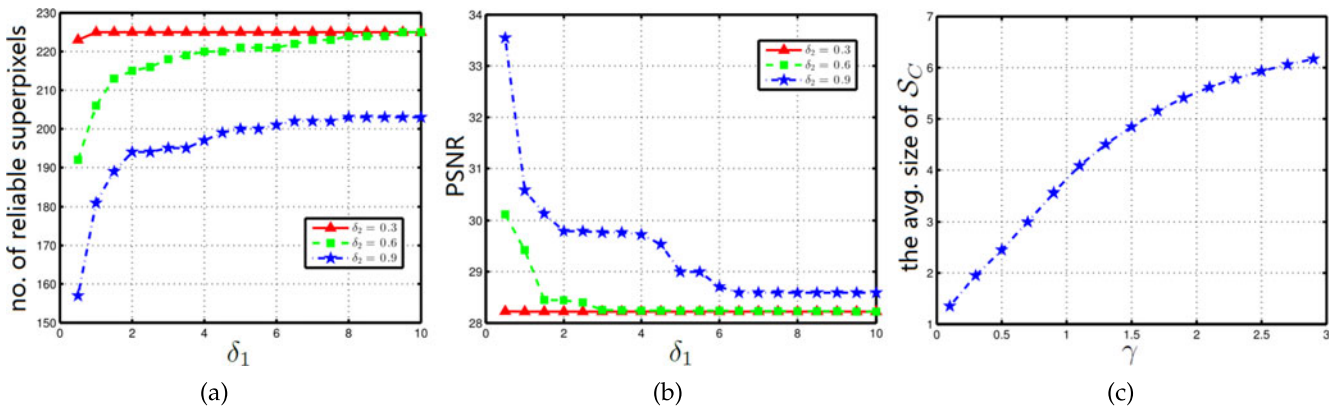


Fig. 11. (a) δ_1 in Eq. (2) checks if a pair of pixel correspondences is consistent, and δ_2 in Eq. (4) checks if a superpixel is reliable. A larger δ_1 (or a smaller δ_2) brings more reliable superpixels. (b) However, a larger δ_1 (or a smaller δ_2) also means more superpixels are misclassified as reliable, reducing the PSNR of the image transformed from the original image by the computed reliable homographies. (c) γ in Section 3.3.2 checks if a reliable superpixel should be associated to an unreliable superpixel. With the increase of γ, the size of S_C will also be increased, where S_C is the set of reliable superpixels associated to an unreliable superpixel.

Fig. 12. Comparison between Google StreetView [11] and our method. The results of StreetView are more blurred than ours.

δ_1 and three values for δ_2. At the other extreme, with the increase of δ_1 (or decrease of δ_2), more superpixels that are actually unreliable are misclassified as reliable, leading to many artifacts when transforming the source image by the homographies estimated from homography fitting. Please refer to Fig. 11b where we used Peak Signal-to-Noise Ratio (PSNR) to measure the artifacts, and smaller PSNR means more artifacts. We found that $\delta_1 = 2$ and $\delta_2 = 0.9$ maintains a proper number of reliable superpixels. We used them as default values in all our experiments.

In the second stage, the parameter γ is a threshold that controls the size of S_C which denotes the set of reliable superpixels associated to an unreliable superpixel. We varied γ and carried out 15 experiments on the dataset of the first column of Fig. 10, and reported the average size of S_C. The results are illustrated in Fig. 11c. 3 to 5 reliable superpixels are enough for the estimation of the homography of an unreliable superpixel, and we set γ as 1.5 in experiments.

In the third stage, the behavior of the warping formulation is controlled by four weights ω_{data}, ω_{sim}, ω_{ht} and ω_{hp} in Eq. (7). The roles of those parameters have been described in the corresponding section. The results of different settings of those parameters are illustrated in Fig. 8.

4.3 Comparison with Previous Methods

Comparison with Google StreetView. Fig. 12 shows the comparisons between Google StreetView [11] and our method. We did experiments on images of two scenes directly downloaded from Google map. One scene (the left of Fig. 12) is a busy city street with buildings on its two sides, and there are many cars parking on the street. Another scene (the right of Fig. 12) comes from a country road with trees, cars, and houses. In both of the scenes, cameras moved a long distance (more than 5 meters), yielding very wide baseline between sampled images. We show the results of Google StreetView and ours side by side. It can be seen that StreetView's results are more blurred than ours. One possible reason we think is as follows. As stated in [11], Google StreetView acquired depth information by laser scanners. With the depth information, a simple but useful 3D popup model can be constructed for an image, based on which any intermediate image is achieved by alpha blending between two 3D models. However, the simple alpha blending may bring severe blurring artifacts. By contrast, our method does not require additional depth information. The accurate homographies between two images computed by our method make the results more clear. This comparison shows that our method can fulfill the task of street view exploration, not only improving the immersive exploration experience, but also reducing the need for expensive equipment.

Comparison with DiscreteFlow. Fig. 13 shows the comparisons between DiscreteFlow [8] and our method. We compared with DiscreteFlow as it is among the 10 best large displacement optical flow methods evaluated on the KITTI benchmark [2] in the writing of this paper. Although a few methods [22], [23] outperform DiscreteFlow, the improvements are not very significant. So, DiscreteFlow is a good representative. With the code provided by the authors of DiscreteFlow, we computed optical flow between a pair of images, and used the rendering algorithm in Section 3.5 of this paper to interpolate intermediate images. The first row of Fig. 13 shows results of DiscreteFlow, while the second row shows our results. From the first and third columns, we can see that it is hard for DiscreteFlow to match foreground objects where optical flow is discontinuous. In the second column, although both methods can handle objects such as cars, trees and houses, DiscreteFlow produces more artifacts in the region of textureless sky. The performance of DiscreteFlow is dramatically reduced by the large occlusion and textureless regions in the wide-baseline examples, while our method can handle these examples very well.

Fig. 13. Comparison between DiscreteFlow [8] and our method. DiscreteFlow cannot handle foreground objects that undergo large displacements, and also cannot handle large textureless regions. Our method can handle the two cases very well.

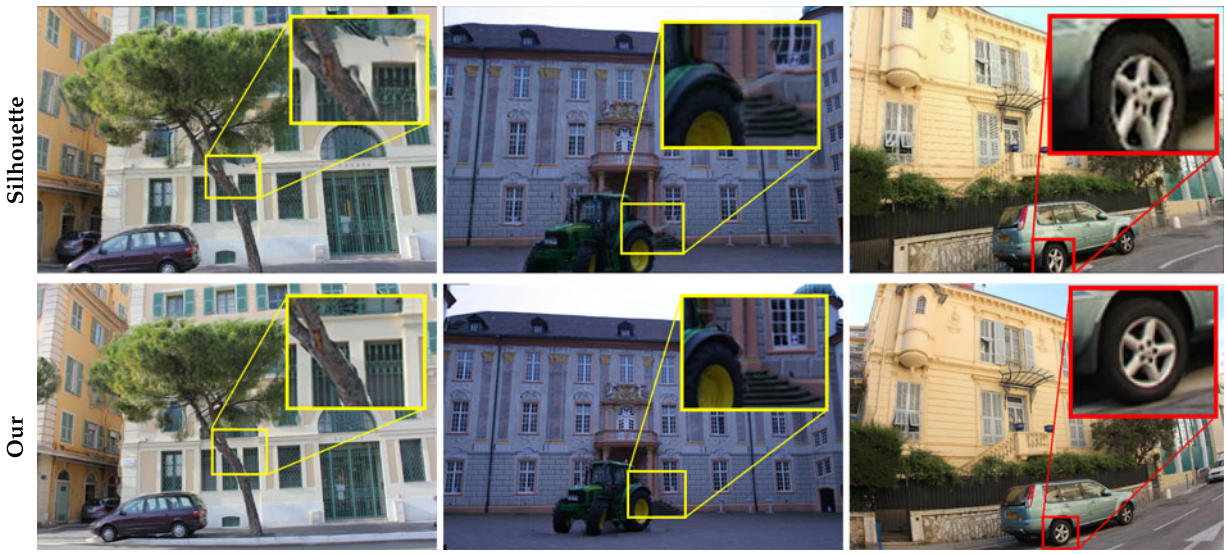


Fig. 14. Comparison between "silhouette" [12] and our method. Visually, our method produces much fewer distortions.

Comparison with IBR Methods. We compared our method with two IBR algorithms: "silhouette-aware warping" [12] and "depth synthesis" [13]. The two approaches were specifically designed for wide-baseline street images with complex objects such as bushes, trees, cars and buildings. However, compared with our approach, these two methods have two disadvantages. First, they demand a large collection of images of the same scene captured from different viewpoints, while our method takes only two images as input which is more flexible. Second, the two IBR methods rely on 3D reconstruction. However, 3D reconstruction is computationally complex. More importantly, as indicated in [1], 3D reconstruction is usually not applicable for KITTI images captured in rural areas due to challenging lighting conditions, vegetation, and image noise, while our method works for both city building scenes and rural scenery scenes. We compared our method with the two IBR methods on their provided dataset in Figs. 14 and 15. Visually, our approach outperforms "silhouette". As shown in Fig. 14, "silhouette" produces more distortions in regions of tree trunk, pillar, and wheel than our method. Our approach outperforms "depth synthesis" when the dataset is not completely fronto-parallel. For example, in the left example of Fig. 15, "depth synthesis" produces distortions in the novel view, especially on the side of the car. That is because "depth synthesis" corrects wrong depth by copying and pasting reliable depth. However, the side of the car is on a slope plane whose depth varies everywhere. Thus, depth copying on the car is invalid. However, since the superpixels on the side of the car belong to the same slope plane, our homography propagation is valid, and can produce more realistic result. However, when facing the fronto-parallel dataset on the right of Fig. 15, both "depth synthesis" and our approach obtain comparatively good results.

Comparison with DeepStereo. We also compared our method with recently developed novel view synthesis method DeepStereo [1] which is based on deep learning techniques. After trained on a large set of Google StreetView or KITTI images, DeepStereo learned the relationships between camera poses, pixel colors and depths under the framework of plane-sweep stereo. Then, given any new pose of a virtual camera, the colors and depths of pixels can be predicted, which were finally weighted summed to obtain novel views. Overall, DeepStereo produces impressive results, and demonstrates the strong strength of deep learning techniques. However, compared with our method, DeepStereo is more heavyweight and cumbersome to be used. First, as a common drawback of deep learning, DeepStereo is computationally intensive. It takes about 12 minutes on a multicore workstation to produce a single newly synthesized image, which is too slow to be deployed on consumer devices at least at its current form. In contrast, our method can render intermediate views in real time. Second, the fact that DeepStereo requires camera poses limits the scope of application of DeepStereo. Only professional dataset such as Google StreetView and KITTI can provide precise camera poses which were usually obtained by

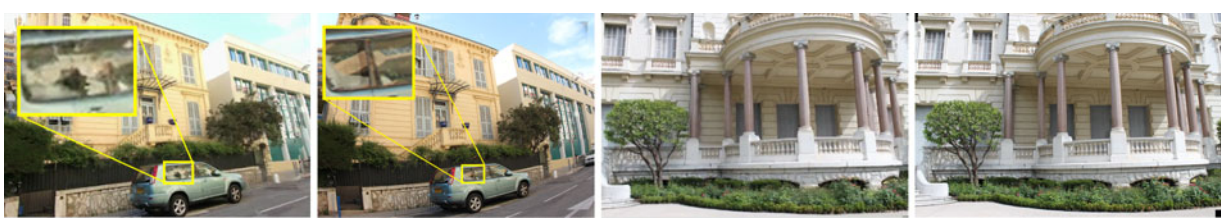


Fig. 15. Comparison between "depth synthesis" [13] and our method. On the left: when the dataset is not fronto-parallel, our method outperforms "depth synthesis". On the right: when the dataset is fronto-parallel, our result matches that of "depth synthesis".

Fig. 16. Comparison between an optical flow algorithm [42], DeepStereo [1] and our method. From left to right are results of the optical flow algorithm, StreetView-trained DeepStereo, KITTI-trained DeepStereo, and our method. From top to bottom are results of different baselines from 1.6 to 4.8 m. Our result matches that of DeepStereo when baseline is 1.6 m. With the increase of baseline, both DeepStereo and our method produce some artifacts: DeepStereo produces more and more blurring, while our method presents ghosting. However, both DeepStereo and our method outperform the traditional optical flow method which was not designed for wide-baseline images.

expensive laser scanners, while the massive images captured by amateur users have no camera pose at all. Thus DeepStereo cannot process the amateur dataset which exists more widely. Our method that takes only two images as input is more flexible to be used on both professional and amateur images.

Although our method is more lightweight and flexible than DeepStereo, it can produce results which are as good as those of DeepStereo. From left to right, Fig. 16 shows the view synthesis results of the optical flow method [42], DeepStereo [1], and our method. From top to bottom, we present different examples with baselines from 1.6 to 4.8 m. DeepStereo and our method produce good results when the baseline is 1.6 m. With the increase of the baseline, both DeepStereo and our method make some artifacts in the synthesized views, where DeepStereo produces more and more blurring on the boundary of the synthesized images while our method produces some ghosting artifacts due to the mismatching between forward and backward homographies. However, both DeepStereo and our method outperform the traditional optical flow method that was not designed for wide-baseline images.

4.3.1 *Limitations*

Our method performs a little worse than previous small-baseline algorithms on small-baseline images. One reason is that the piecewise homography assumption is too rough to search precise pixel-to-pixel correspondences. Small errors of those correspondences may bring apparent artifacts. However, such artifacts are not apparent in wide-baseline interpolation as they are outweighed by large object transformations. Another disadvantage is that our approach cannot generate free-viewpoint walkthroughs compared to DeepStereo and IBR methods, which is the common shortcoming of 2D methods. Third, although our method can handle very wide-baseline images to some extent, it would still fail to find correspondences between objects if they appear only in one input image, which may yield holes in the interpolated images.

5 Conclusion and Future Work

We have presented a lightweight and effective wide-baseline image interpolation algorithm. We creatively combined homography fitting and homography propagation together, and showed that the combination can dramatically increase

the accuracy of homographies of superpixels of an image with respect to another distant image. The combination was facilitated by discriminating reliable superpixels from unreliable ones based on the consistency of Appropriate Nearest Neighbor Field, and was further compensated by a homography-constrained warping optimization. Although each step of the proposed method used some existing techniques, our contribution is the three-step algorithm as a whole which demonstrates that simple homography computing and refining operators can fulfill complex task of wide-baseline image interpolation. We have conducted various kinds of experiments on both city and rural scenes which show that our algorithm is indeed more lightweight than previous method, but achieves results at least as good as those of previous heavyweight approaches.

Currently, the proposed algorithm deals with a pair of images at a time, which may be inconsistent when processing a sequence of street images. Thus a future work is to extend the algorithm to handle multiple images at a time, though may bring more computational load.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This work was partly supported by the NSFC (No. 61602183, 61379087, and 61572202), UGC grant for research (No. 4055060), China Postdoctoral Science Foundation Grant (No. 2016M590780), and the Key Project of Natural Science Foundation of Guangdong, China (No. S2013020012795).

References

[1] J. Flynn, I. Neulander, J. Philbin, and N. Snavely, "DeepStereo: Learning to predict new views from the world's imagery," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 5515–5524.

[2] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The kitti vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

[3] A.-M. Huang and T. Nguyen, "Correlation-based motion vector processing with adaptive interpolation scheme for motion-compensated frame interpolation," *IEEE Trans. Image Process.*, vol. 18, no. 4, pp. 740–752, Apr. 2009.

[4] S.-G. Jeong, C. Lee, and C.-S. Kim, "Motion-compensated frame interpolation based on multihypothesis motion estimation and texture optimization," *IEEE Trans. Image Process.*, vol. 22, no. 11, pp. 4497–4509, Nov. 2013.

[5] D. Mahajan, F.-C. Huang, W. Matusik, R. Ramamoorthi, and P. Belhumeur, "Moving gradients: A path-based method for plausible image interpolation," *ACM Trans. Graph.*, vol. 28, no. 3, 2009, Art. no. 42.

[6] T. Stich, C. Linz, G. Albuquerque, and M. Magnor, "View and time interpolation in image space," *Comput. Graph. Forum*, vol. 27, no. 7, pp. 1781–1787, 2008.

[7] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *Int. J. Comput. Vis.*, vol. 92, no. 1, pp. 1–31, 2011.

[8] M. Menze, C. Heipke, and A. Geiger, "Discrete optimization for optical flow," in *Pattern Recognition*. Berlin, Germany: Springer, 2015, pp. 16–28.

[9] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *Proc. 12th Eur. Conf. Comput. Vis.*, Oct. 2012, pp. 611–625.

[10] C. Verleysen, T. Maugey, P. Frossard, and C. De Vleeschouwer, "Wide-baseline object interpolation using shape prior regularization of epipolar plane images," Inst. Elect. Electron. Eng., Piscataway, NJ, USA, Tech. Rep. EPFL-ARTICLE-200492, 2015.

[11] D. Anguelov, et al., "Google street view: Capturing the world at street level," *Computer*, vol. 43, no. 6, pp. 32–38, 2010.

[12] G. Chaurasia, O. Sorkine, and G. Drettakis, "Silhouette-aware warping for image-based rendering," *Comput. Graph. Forum*, vol. 30, no. 4, pp. 1223–1232, 2011.

[13] G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis, "Depth synthesis and local warps for plausible image-based navigation," *ACM Trans. Graph.*, vol. 32, no. 3, 2013, Art. no. 30.

[14] B. K. Horn and B. G. Schunck, "Determining optical flow," in *Proc. Tech. Symp. East Int. Soc. Optics Photonics*, 1981, pp. 319–331.

[15] T. Stich, C. Linz, C. Wallraven, D. Cunningham, and M. Magnor, "Perception-motivated interpolation of image sequences," *ACM Trans. Appl. Perception*, vol. 8, no. 2, 2011, Art. no. 11.

[16] W. Enkelmann, "Investigations of multigrid algorithms for the estimation of optical flow fields in image sequences," *Comput. Vis. Graph. Image Process.*, vol. 43, no. 2, pp. 150–177, 1988.

[17] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani, "Hierarchical model-based motion estimation," in *Proc. 2nd Eur. Conf. Comput. Vis.*, 1992, pp. 237–252.

[18] T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 3, pp. 500–513, Mar. 2011.

[19] Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu, "Large displacement optical flow from nearest neighbor fields," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 2443–2450.

[20] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "DeepFlow: Large displacement optical flow with deep matching," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 1385–1392.

[21] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "EpicFlow: Edge-preserving interpolation of correspondences for optical flow," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1164–1172.

[22] C. Bailer, B. Taetz, and D. Stricker, "Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 4015–4023.

[23] J. Yang and H. Li, "Dense, accurate optical flow estimation with piecewise parametric model," in *Proc. 31st IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1019–1027.

[24] L. Bao, Q. Yang, and H. Jin, "Fast edge-preserving PatchMatch for large displacement optical flow," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 3534–3541.

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

[26] M. Wang, X.-J. Zhang, J.-B. Liang, S.-H. Zhang, and R. R. Martin, "Comfort-driven disparity adjustment for stereoscopic video," *Comput. Visual Media*, vol. 1, no. 2, pp. 3–17, 2016.

[27] M. Galun, T. Amir, T. Hassner, R. Basri, and Y. Lipman, "Wide baseline stereo matching with convex bounded distortion constraints," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2228–2236.

[28] C. Lipski, F. Klose, and M. Magnor, "Correspondence and depth-image based rendering a hybrid approach for free-viewpoint video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 6, pp. 942–951, Jun. 2014.

[29] J. Yang, S. E. Reed, M.-H. Yang, and H. Lee, "Weakly-supervised disentangling with recurrent transformations for 3D view synthesis," in *Proc. Advances Neural Inf. Process. Syst.*, 2015, pp. 1099–1107.

[30] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros, "View synthesis by appearance flow," in *Proc. 14th Eur. Conf. Comput. Vis.: Part IV*, (Eds.), B. Leibe, J. Matas, N. Sebe, and M. Welling, 2016, pp. 286–301.

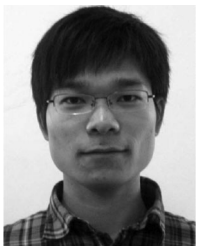
[31] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

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

[33] 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, 2011, Art. no. 70.

[34] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

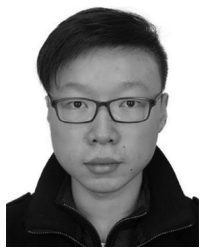
[35] M. Grundmann, V. Kwatra, M. Han, and I. Essa, "Efficient hierarchical graph-based video segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 2141–2148.
[36] Z. Zhu, R. R. Martin, and S.-M. Hu, "Panorama completion for street views," *Comput. Visual Media*, vol. 1, no. 1, pp. 49–57, 2015.
[37] F. Liu, M. Gleicher, H. Jin, and A. Agarwala, "Content-preserving warps for 3D video stabilization," *ACM Trans. Graph.*, vol. 28, no. 3, 2009, Art. no. 44.
[38] G.-X. Zhang, M.-M. Cheng, S.-M. Hu, and R. R. Martin, "A shape-preserving approach to image resizing," *Comput. Graph. Forum*, vol. 28, no. 7, pp. 1897–1906, 2009.
[39] X.-Y. Li, T. Ju, and S.-M. Hu, "Cubic mean value coordinates," *ACM Trans. Graph.*, vol. 32, no. 4, 2013, Art. no. 126.
[40] J. Nocedal and S. Wright, *Numerical Optimization*. Berlin, Germany: Springer, 2006.
[41] P. J. Burt and E. H. Adelson, "A multiresolution spline with application to image mosaics," *ACM Trans. Graph.*, vol. 2, no. 4, pp. 217–236, 1983.
[42] C. Liu, "Beyond pixels: Exploring new representations and applications for motion analysis," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, 2009.



Yongwei Nie received the BSc and PhD degrees from the Computer School, Wuhan University, in 2009 and 2015, respectively. Currently, he is an assistant professor in the School of Computer Science & Engineering, South China University of Technology. His research interests include image and video editing and computational photography.

Zhensong Zhang received the BE degree from Xidian University, in 2011, and the MS degree from the University of Chinese Academy of Sciences, in 2014. He is working toward the PhD degree in the Department of Computer Science & Engineering, The Chinese University of Hong Kong. His research interests include image and video editing and computational photography.

Hanqiu Sun received the MS degree in electrical engineering from the University of British Columbia and the PhD degree in computer science from the University of Alberta, Canada. She is an associate professor with The Chinese University of Hong Kong. Her research interests include virtual reality, interactive graphics/animation, real-time hypermedia, virtual surgery, mobile image/video synopsis and navigation, touch-enhanced, and dynamics simulations. Contact her at hanqiu@cse.cuhk.edu.hk. She is a member of the IEEE.



Tan Su received the BEng degree from The Chinese University of Hong Kong, in 2015. He is working toward the Mphil degree in the Department of Computer Science & Engineering, The Chinese University of Hong Kong. His research interests include image and video editing and computational photography.

Guiqing Li received the BSc degree from the University of Science and Technology of China, in 1987, the MSc degree from Nankai University, Tianjin, in 1990, and the PhD degree from the Institute of Computing Technology, Beijing, in 2001. He is a professor in the College of Computer Science and Engineering, South China University of Technology (SCUT). Before joining SCUT in September 2003, he worked as a postdoctoral researcher in the State Key Lab of CAD&CG, Zhejiang University.