Interactive Video Completion

Makoto Okabe, Keita Noda, Yoshinori Dobashi, *Member, IEEE,* and Ken Anjyo, *Member, IEEE*

Abstract—We propose an interactive video completion method aiming for practical use in a digital production workplace. The results of earlier automatic solutions often require considerable amount of manual modifications to make them usable in practice. To reduce such a laborious task, our method offers an efficient editing tool. Our iterative algorithm estimates the flow fields and colors in space-time holes in the video. As in earlier approaches, our algorithm uses an L^1 data term to estimate flow fields. However, we employ a novel L^2 data term to estimate temporally coherent color transitions. Our GPU implementation enables the user to interactively complete a video by drawing holes and immediately removes objects from the video. In addition, our method successfully interpolates sparse modifications initialized by the designer. According to our subjective evaluation, the videos completed with our method look significantly better than those with other state-of-the-art approaches.

Index Terms—Image and video inpainting, interactive technique, subjective evaluation, optical flow, image-based rendering.

1 Introduction

Video completion is one of the most highly demanded skills in the digital production workplace, because postproduction designers must always complete tasks such as the removal of objects, logos, annotations, and noises from videos. As there are few practical methods for video completion, these tasks rely heavily on designers' manual edits to create convincing natural-looking results. However, such manual video completion is very demanding. Hence, we set out to develop a practical method to help designers.

A lot of practical methods have been proposed for image completion rather than video completion, some of which are widely implemented in commercial image editing software products. State-of-the-art image completion algorithms are computationally efficient, thus allowing the users to work interactively through trial and error: this is crucial to obtain satisfactory results, because the image automatically completed by a computer is often far from perfect.

However, video completion remains challenging for two reasons:

1) Computational complexity increases with the number of video frames, which makes it difficult for the user to work interactively by trial and error.
2) The quality of the completed video is often not satisfactory due to failures in estimating the dynamic motion of the camera and objects.

For example, the previous methods developed by [1], [2], [3], [4], [5], [6], [7] suffer from these two problems: they are computationally costly and their results contain a significant number of visible artifacts, caused by the loss of temporal coherence. Bokov et al. proposed an efficient method that is more than 100 times faster than the previous method [8], which takes 75 seconds to complete a 90-frame 'camel' video with a resolution of 854×480 pixels. Murase et al. recently proposed a more efficient method that completes a video with a resolution of 832×448 pixels in video rate, e.g., 32 frames per second (FPS) [9]. However, visible artifacts are often found in the results produced by these efficient algorithms. Furthermore, such artifacts are difficult to modify manually. Note that given a completed video with such artifacts, digital artists have to remove them by typically performing manual image editing frame by frame.

To address these issues, we propose a novel algorithm based on a flow-guided color estimation approach, which we demonstrate is useful for practical usage (Fig. 1). We present three technical contributions:

1) Our iterative algorithm estimates flow fields and colors in the holes alternately. We propose to use the L^1 data term for flow field estimation but the L^2 data term for color estimation. This is a technically simple modification, but the quality of the completed video is significantly improved: the L^2 data term successfully removes temporally visible artifacts caused by the L^1 data term (Fig. 1-a and 1-b).
2) Our L^2 data term also allows the user to modify the completed video efficiently. We demonstrate that the manual modification on single or a few frames yields dramatic improvements in the quality of the completed video. It is impossible to achieve such improvements using the L^1 data term (Fig. 1-c and 1-d).
3) Our method is computationally efficient enough for interactive video completion. The user interactively draws a mask and our method immediately removes objects from the video. Such a user interaction has never been demonstrated in the previous papers.

- *M. Okabe and K. Noda are with Graduate School of Engineering, Shizuoka University, 3-5-1 Johoku, Naka-ku, Hamamatsu, Shizuoka, 432-8561, Japan.*
  E-mail: m.o@acm.org, noda.keita.17@shizuoka.ac.jp
- *Y. Dobashi is with Graduate School of Information Science and Technology, Hokkaido University, Kita-ku, Kita 14, Nishi 9, Sapporo, 060-0814, Japan.*
  E-mail: doba@ime.ist.hokudai.ac.jp
- *K. Anjyo is with OLM Digital, Inc. and CMIC, Victoria University of Wellington, Dens Kono Bldg., Room 302 OLM Digital ViPlus, 1-8-8 Wakabayashi, Setagaya, Tokyo, 154-0023, Japan.*
  E-mail: anjyo@acm.org

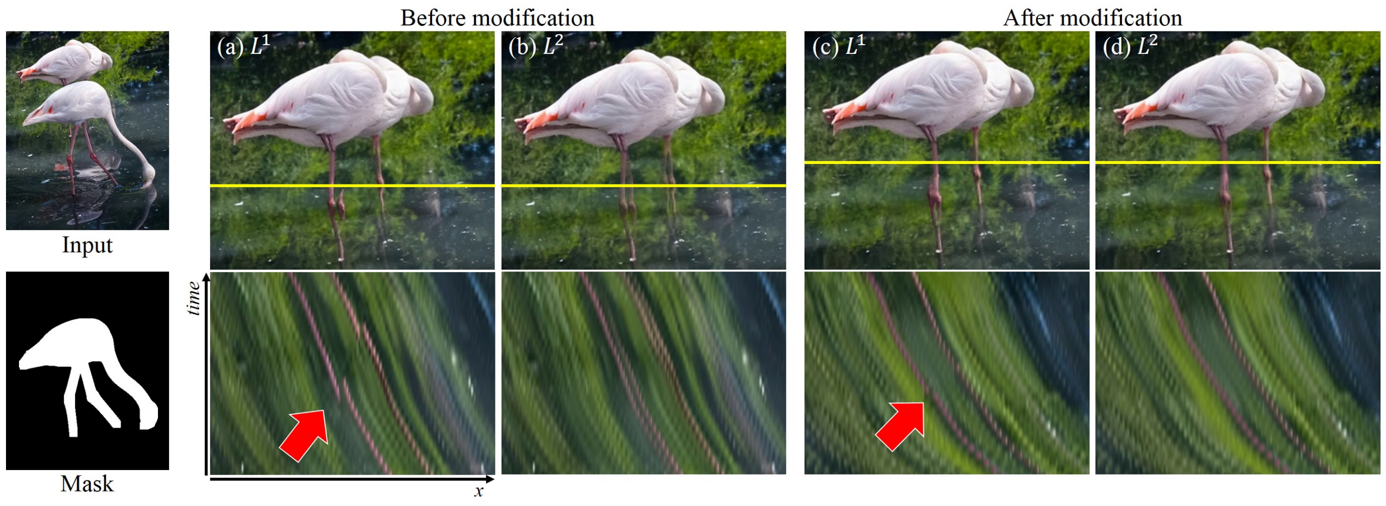

Fig. 1. We removed the flamingo specified by the mask from the input video. The time required to process a 80-frame 'flamingo' video at a pixel resolution of 854×480 was 4.3 seconds. The top row shows frames from the completed videos and the bottom row shows the corresponding $x - time$ slice along the yellow line. The result based on the L^1 data term has artifacts (a), which are clearly visible as seams in the $x - time$ slice. On the other hand, our result based on the L^2 data term has smoother color transitions (b): our result still suffers from ghosting artifacts but was more desirable for the audience (see Sec. 4.8). To remove the artifacts, we manually modified a frame and reapplied our algorithm. The result based on the L^1 data term still contains artifacts (c), but artifacts were successfully removed from the result based on the L^2 data term (d). Our supplementary video shows the differences more clearly.

We demonstrate the successful completion of a variety of videos using our prototyping system. According to our subjective evaluation, the quality of the videos completed by our method is significantly better than those completed using state-of-the-art methods. The result of our subjective evaluation clearly indicates that our L^2 scheme proves to be superior to the L^1 scheme of all the other methods.

2 Previous Work

There are two main approaches to automatic video completion: one is the patch-based approach and the other is the flow-guided color estimation approach.

Wexler et al. proposed the first patch-based method for video completion [1]. They used a $5 \times 5 \times 5$ space-time patch and iterated a pair of operations: 1) nearest neighbor patch searches and 2) calculation of the weighted averages to fill in the holes. They showed that the problem of video completion can be formalized as a maximum likelihood estimation and their iterative optimization is an expectation maximization (EM)-like algorithm. Newson et al. extended Wexler et. al.'s method by introducing PatchMatch and developing an efficient algorithm [4].

To improve temporal coherence of completed videos, not only colors but also flow fields in space-time holes are estimated [10], [11]. Le et al. extended Newson et al.'s algorithm by taking flow fields into account [7]. They did not use a cube-shaped patch, but instead chose a patch with $x - y$ slices that are varied based on the flow field. This improves the quality of the completed video dramatically compared with the previous methods and often produces the best results.

Roxas et al. proposed a flow-guided color estimation method [5]. They introduced an energy function for video completion by extending the energy function used for optical flow estimation. Huang et al. proposed an energy function that combines the patch-based approach with flow-guided color estimation [6]. They proposed an iterative optimization algorithm to minimize the energy function, where the patch-based approach, flow-guided color estimation, and flow field estimation are iterated until they converge. However, the computational complexity of this optimization algorithm is high, and it takes 3 hours to complete a 90-frame video with a resolution of 854×480 pixels, thus prohibiting practical usage. Bokov et al. proposed an efficient method that is more than 100 times faster than Huang et al.'s method [8]. As the bottleneck of the flow-guided color estimation is the optical flow estimation, Bokov et al. efficiently estimated flow fields by applying a fast optical flow algorithm to a sparse grid in the image space [12]. This successfully reduced the computational complexity. Instead of an iterative optimization algorithm to minimize the energy function, the flow-guided color estimation is solved only once. Recently, Murase et al. more efficiently estimated flow fields of occluded background regions using convolutional neural networks (CNNs) [9]. This method completes a video with a resolution of 832×448 pixels in video rate, e.g., 32 FPS. These methods work well for a variety of videos, but the completed videos tend to contain visible artifacts, especially space-time seams in dynamic regions. To address these issues, we propose a fast flow-guided color estimation method that produces significantly better results.

The approach of estimating colors and flow fields at the same time is applied to other video editing problems: for example, Nandoriya et al. removed reflections from a video [13]. The propagation of colors along the known flow fields of a video is another issue: Sadek et al. propagated the user's edit on a frame throughout the video with less noticeable artifacts [14].

3 Flow-guided Color Estimation

Our goal is to complete space-time holes in videos and produce natural-looking results. Let $\mathbf{V}$ denote the video and W, H, and T be its width, height, and number of frames, i.e., $\mathbf{V}$ is the space-time volume with dimensions $W \times H \times T$ voxels. Let $\mathbf{p}$ be the voxel position (x, y, t). $\mathbf{V}(\mathbf{p})$ gives the voxel value at $\mathbf{p}$, which corresponds to a scalar intensity value from 0 to 1. Let $\mathbf{F}_f$ denote the forward flow field of $\mathbf{V}$. The flow vector at $\mathbf{p}$ is $\mathbf{F}_f(\mathbf{p}) = (dx, dy, 1)$, which indicates that a voxel position $\mathbf{p} = (x, y, t)$ at t-th frame moves to $\mathbf{p} + \mathbf{F}_f(\mathbf{p}) = (x + dx, y + dy, t + 1)$ at $(t + 1)$-th frame. Similarly, let $\mathbf{F}_b$ denote the backward flow field, i.e., $\mathbf{F}_b(\mathbf{p}) = (dx, dy, -1)$. Let $\mathbf{P} = \{(x, y, t) | 1 \leq x \leq W, 1 \leq y \leq H, 1 \leq t \leq T\}$ be the set of voxel positions in the space-time volume. Let $\mathbf{H}$ be the set of voxel positions in the space-time holes in $\mathbf{V}$. We describe the naturalness of a completed video using the following energy function:

$$E = E_f + E_b, \tag{1}$$

where

$$E_f = \sum_{\mathbf{p} \in \mathbf{P}} w_f(\mathbf{p}) \lambda |\mathbf{V}(\mathbf{p}) - \mathbf{V}(\mathbf{p} + \mathbf{F}_f(\mathbf{p}))| + |\nabla \mathbf{F}_{f,x}(\mathbf{p})| + |\nabla \mathbf{F}_{f,y}(\mathbf{p})|, \tag{2}$$

and

$$E_b = \sum_{\mathbf{p} \in \mathbf{P}} w_b(\mathbf{p}) \lambda |\mathbf{V}(\mathbf{p}) - \mathbf{V}(\mathbf{p} + \mathbf{F}_b(\mathbf{p}))| + |\nabla \mathbf{F}_{b,x}(\mathbf{p})| + |\nabla \mathbf{F}_{b,y}(\mathbf{p})|. \tag{3}$$

$w_f(\mathbf{p})$ and $w_b(\mathbf{p})$ are weights on the data terms used when computing the flow fields. Each of $\mathbf{F}_{*,x}$ and $\mathbf{F}_{*,y}$ represents x or y component of $\mathbf{F}_*$ respectively, and ∇ represents the gradient operator $(\partial/\partial x, \partial/\partial y)$. We want to compute $\mathbf{V}$, $\mathbf{F}_f$, and $\mathbf{F}_b$ that minimize E, i.e., we want to solve $\underset{\mathbf{V}, \mathbf{F}_f, \mathbf{F}_b}{\arg\min} E$. We set λ to 0.1 in all of our experiments.

The first terms of Eqs. 2 and 3 are the data terms representing the similarity between the voxel values of adjacent frames. The other terms are smoothness terms representing the smoothness of the forward and backward flow fields. Eqs. 2 and 3 are all derived from the energy function of the optical flow estimation, which consists of an L^1 data term and the sum of the total variation (TV) of the flow field [15].

The energy function of Eq. 1 is non-convex and may have an infinite number of local minima. To complete the video with a good local minimum, we begin with an initial video, and estimate $\mathbf{V}$ iteratively by refining this video to decrease the energy function. We show our iterative algorithm in Algorithm 1.

We compute the initial flow fields $\mathbf{F}_f$ and $\mathbf{F}_b$ (line 2 in Algorithm 1). We set $w_f(\mathbf{p})$ and $w_b(\mathbf{p})$ as follows:

$$w_f(\mathbf{p}) = \begin{cases} 0 & \mathbf{p} \in \mathbf{H} \quad or \quad \mathbf{p} + \mathbf{F}_f(\mathbf{p}) \in \mathbf{H}, \\ 1 & otherwise, \end{cases} \tag{4}$$

$$w_b(\mathbf{p}) = \begin{cases} 0 & \mathbf{p} \in \mathbf{H} \quad or \quad \mathbf{p} + \mathbf{F}_b(\mathbf{p}) \in \mathbf{H}, \\ 1 & otherwise. \end{cases} \tag{5}$$

We then compute the flow fields by applying the TV-L^1 optical flow estimation algorithm [15]. Since the data terms are ignored in the holes ($\mathbf{H}$), the flows estimated outside the holes ($\overline{\mathbf{H}}$) are interpolated into the holes ($\mathbf{H}$). In the TV-L^1 optical flow estimation algorithm, an iterative optimization algorithm is applied to the relaxed version of the problem, where the data and smoothness terms are minimized alternately until they converge. We apply the TV-L^1 algorithm independently for forward and backward flow fields: we minimize Eq. 2 to initialize $\mathbf{F}_f$; we minimize Eq. 3 to initialize $\mathbf{F}_b$.

Algorithm 1 Our iterative algorithm

```
Input: $\mathbf{V}$: an input video,
       $\mathbf{H}$: a set of voxel positions in the space-time holes
Output: $\mathbf{V}$: a completed video
 1: Downsample $\mathbf{V}$ to $(L-1)$-th pyramid level
 2: Initialize $\mathbf{F}_f$ and $\mathbf{F}_b$
        by setting $w_f(\mathbf{p})$ and $w_b(\mathbf{p})$ based on Eqs. 4 and 5
        and applying TV-$L^1$ algorithm
 3: for $l$ from $L-1$ to 0 do
 4:     Set $w_f(\mathbf{p})$ and $w_b(\mathbf{p})$ based on Eqs. 6 and 7
 5:     for $i$ from 1 to $3^l$ do
 6:         Fix $\mathbf{F}_f$ and $\mathbf{F}_b$ and update $\mathbf{V}$
              by minimizing Eq. 10 for all $\mathbf{p} \in \mathbf{H}$
 7:         Fix $\mathbf{V}$ and update $\mathbf{F}_f$ and $\mathbf{F}_b$
              by applying TV-$L^1$ algorithm
 8:     end for
 9:     Upsample $\mathbf{V}$, $\mathbf{F}_f$, and $\mathbf{F}_b$.
10: end for
```

During each iteration (from line 5 to line 8 in Algorithm 1), we alternate between the completed video ($\mathbf{V}$) and the flow fields ($\mathbf{F}_f$ and $\mathbf{F}_b$) as the variable with respect to which E is minimized. We first fix $\mathbf{F}_f$ and $\mathbf{F}_b$ and minimize E to update $\mathbf{V}$. We are interested only in the data terms here, which can be minimized using the algorithms applied in previous methods [5], [6], [8]. However, the results produced by the previous methods contain significant visible artifacts, since minimization of the L^1 data terms brings temporally incoherent solutions. To address this issue, we estimate the voxel value by minimizing E based on the L^2 data terms rather than the L^1 data terms: note that the solution using the L^2 data terms is one of the solutions of the L^1 data terms (as described in Sec. 3.2.1), but our results have much less artifacts. We then fix $\mathbf{V}$ and minimize E to update $\mathbf{F}_f$ and $\mathbf{F}_b$. We set $w_f(\mathbf{p})$ and $w_b(\mathbf{p})$ as follows:

$$w_f(\mathbf{p}) = \begin{cases} \alpha & \mathbf{p} \in \mathbf{H} \quad or \quad \mathbf{p} + \mathbf{F}_f(\mathbf{p}) \in \mathbf{H}, \\ 1 & otherwise, \end{cases} \tag{6}$$

$$w_b(\mathbf{p}) = \begin{cases} \alpha & \mathbf{p} \in \mathbf{H} \quad or \quad \mathbf{p} + \mathbf{F}_b(\mathbf{p}) \in \mathbf{H}, \\ 1 & otherwise, \end{cases} \tag{7}$$

where α is the user-specified parameter. We then apply the TV-L^1 optical flow estimation algorithm. The flow fields are improved with respect to the previous iteration.

We extend this iterative algorithm to the multi-resolutional approach to efficiently obtain the final result (from line 3 to line 10 in Algorithm 1). We start with a coarse version of $\mathbf{V}$ (line 1 in Algorithm 1). $\mathbf{F}_f$ and $\mathbf{F}_b$ are initialized in the coarse resolution. We then use bilinear interpolation to upsample $\mathbf{V}$, $\mathbf{F}_f$, and $\mathbf{F}_b$ and switch to a finer level once the coarser level is finished. We let L

denote the number of pyramid levels. In our experiments, the number of iterations of our iterative algorithm for l-th pyramid level is 3^l, where $l \in \{0, ..., L-1\}$, and 0-th and $(L-1)$-th levels correspond to the finest and coarsest resolutions, respectively. We have chosen 3^l as the number of iterations, since it gives good convergence of our iterative algorithm.

In Sec. 3, we describe our method assuming that the input is a grayscale video, i.e., each voxel of $\mathbf{V}$ has an intensity value. When the input video is multi-channel, e.g., each voxel has red-green-blue (RGB) color channels, we apply the method described in Sec. 3 for each color channel independently. However, since, the flow fields $\mathbf{F}_f$ and $\mathbf{F}_b$ must be the same for all the color channels, we compute the flow fields by applying the TV-L^1 algorithm to the grayscale video whose intensity value is computed as the average of all the color channels.

3.1 Differences from Previous Methods

Our energy function of Eq. 1 is different from any other energy functions used in the previous methods. Roxas et al.'s energy function is similar to ours but they added an energy term to guarantee the temporal smoothness of flow fields. Also, while $w_f(\mathbf{p})$ and $w_b(\mathbf{p})$ are constant in our method, they were unknown variables to be optimized in Roxas et al.'s method. As the result, the minimization required a complex optimization algorithm whose computational cost was high. A part of Huang et al.'s energy function is also similar to ours but they added an energy term to guarantee the similarity between corresponding local patches. They minimized the added term by a patch-based texture synthesis method but this was computationally expensive. Bokov et al. defined no energy function, and they used neither an iterative optimization algorithm nor a multi-resolutional approach. As the result, the method was fast but it produced visible artifacts especially when the camera motion was dynamic.

We design our energy function inspired by these methods but it is more simple compared with the previous methods. Since we compute only $\mathbf{V}$, $\mathbf{F}_f$, and $\mathbf{F}_b$, our optimization algorithm has also become simple that can produce significantly better results more efficiently than the previous methods.

3.2 Minimizing Data Terms

Since only data terms are related to the update of $\mathbf{V}$, we rewrite Eq. 1 as follows:

$$\begin{aligned} E^{data} &= \sum_{\mathbf{p}\in\mathbf{H}} |\mathbf{V}(\mathbf{p}) - \mathbf{V}(\mathbf{p}+\mathbf{F_f}(\mathbf{p}))| \\ &+ \sum_{\mathbf{p}\in\mathbf{H}} |\mathbf{V}(\mathbf{p}) - \mathbf{V}(\mathbf{p}+\mathbf{F_b}(\mathbf{p}))|. \end{aligned} \tag{8}$$

$\mathbf{p}+\mathbf{F}_f(\mathbf{p})$ and $\mathbf{p}+\mathbf{F}_b(\mathbf{p})$ do not often point to a grid point in $W \times H \times T$ voxels. Hence, it is necessary to estimate $\mathbf{V}(\mathbf{p}+\mathbf{F_f}(\mathbf{p}))$ and $\mathbf{V}(\mathbf{p}+\mathbf{F_b}(\mathbf{p}))$ by interpolating the values of the neighboring voxels. As the result, the minimization of Eq. 8 gives a blurry completed video.

To address this issue, the previous methods tracked the forward and backward flow fields and sampled the voxel values. Fig. 2 shows how these tracking and sampling are performed in the $x - time$ slice. Let $\mathbf{p}_0$ be equal to $\mathbf{p}$. We define $\mathbf{p}_u$ so that $\mathbf{p}_{u+1} = \mathbf{p}_u + \mathbf{F}_f(\mathbf{p}_u)$ for $u \geq 0$ and $\mathbf{p}_{u-1} = \mathbf{p}_u + \mathbf{F}_b(\mathbf{p}_u)$ for $u \leq 0$. The method computes the forward and backward trajectories of a particle starting from $\mathbf{p}_0$ until it leaves the holes. Let $\mathbf{p}_F$ and $\mathbf{p}_B$ be the positions of the endpoints of the trajectories: the suffixes F and B denote temporal distances from $\mathbf{p}$ to the endpoints (see Fig. 2).

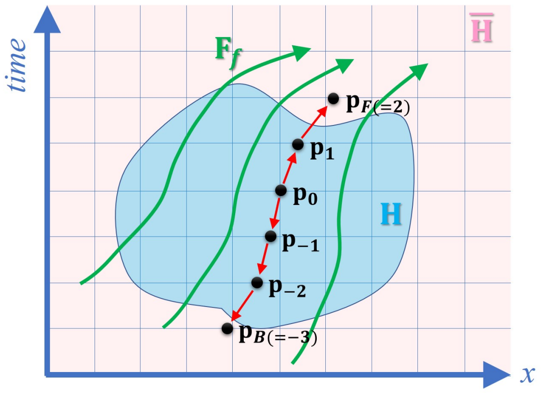

Fig. 2. Starting from $\mathbf{p}_0$, we track the forward and backward flow fields and sample the voxel values at $\mathbf{p}_F$ and $\mathbf{p}_B$.

Roxas et al. [5] and Bokov et al. [8] determined the voxel value $\mathbf{V}(\mathbf{p})$ so that $\mathbf{V}(\mathbf{p}) = \mathbf{V}(\mathbf{p}_F)$ or $\mathbf{V}(\mathbf{p}) = \mathbf{V}(\mathbf{p}_B)$. Since Huang et al. had an initial solution computed by their patch-based approach, $\mathbf{V}(\mathbf{p})$ was iteratively updated. However, unless such an initial solution is given, their method also gives the voxel value near to either $\mathbf{V}(\mathbf{p}_F)$ or $\mathbf{V}(\mathbf{p}_B)$ [6]. All of these methods bring temporally incoherent results.

To address this issue, we compute the weighted average of $\mathbf{V}(\mathbf{p}_F)$ and $\mathbf{V}(\mathbf{p}_B)$ using $\frac{1}{F}$ and $\frac{1}{B}$ as weights. Our method dramatically suppresses artifacts caused by temporal incoherence.

3.2.1 *Mathematical Discussion*

The previous methods use one of $\mathbf{V}(\mathbf{p}_F)$ and $\mathbf{V}(\mathbf{p}_B)$ to determine the voxel value $\mathbf{V}(\mathbf{p})$. This can be considered as the minimization of the following energy function consisting of L^1 terms:

$$E_{\mathbf{p}}^{data,L^1} = \sum_{u=B}^{F-1} |\mathbf{V}(\mathbf{p}_u) - \mathbf{V}(\mathbf{p}_{u+1})|, \tag{9}$$

where we assume $\mathbf{V}(\mathbf{p}_F)$ and $\mathbf{V}(\mathbf{p}_B)$ as known variables and $\{\mathbf{V}(\mathbf{p}_u) | B < u < F\}$ as the set of unknown variables. This energy function is derived from Eq. 8 by removing spatial relations between $\mathbf{p}$ and its neighboring voxels and introducing temporal relations along the trajectory through $\mathbf{p}$. The optimization problem of Eq. 9 can be established independently at each voxel position $\mathbf{p}$. Fig. 3 illustrates the solutions of the minimization of Eq. 9. All the red, green, and blue lines minimize Eq. 9 but the red and blue lines have a temporally sudden change in the voxel values. These cause significant visible artifacts of temporal incoherence but all the previous methods are intended to minimize Eq. 9.

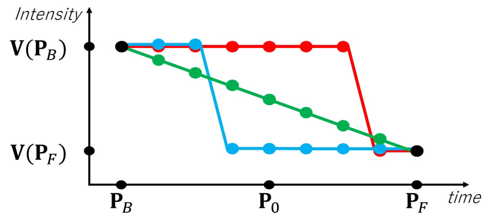

Fig. 3. Each of the red, green, and blue lines is the plot of $\{\mathbf{V}(\mathbf{p}_u)|B < u < F\}$ that minimizes Eq. 9. Only the green line minimizes Eq. 10. The red and blue lines have a sudden change but the green line has the smooth transition.

To address this issue, we minimize the following energy function consisting of not L^1 but L^2 terms:

$$E_{\mathbf{p}}^{data,L^2} = \sum_{u=B}^{F-1} |\mathbf{V}(\mathbf{p}_u) - \mathbf{V}(\mathbf{p}_{u+1})|^2. \tag{10}$$

The minimization of Eq. 10 enforces smooth transitions between $\mathbf{V}(\mathbf{p}_F)$ and $\mathbf{V}(\mathbf{p}_B)$, i.e., only the green line in Fig. 3 can be the solution. $E_{\mathbf{p}}^{data,L^2}$ can be minimized exactly by solving the linear system $\frac{dE_{\mathbf{p}}^{data,L^2}}{d\mathbf{V}} = 0$: this turns out to be equivalent to the weighted average of $\mathbf{V}(\mathbf{p}_F)$ and $\mathbf{V}(\mathbf{p}_B)$ using $\frac{1}{F}$ and $\frac{1}{B}$ as weights.

3.3 Remaining Regions

There often remain voxels whose intensity value has not been estimated by the process described above, because it is impossible to estimate $\mathbf{V}(\mathbf{p})$ when the forward and backward trajectories go out from the space-time volume $\mathbf{V}$ before $\mathbf{p}_F$ and $\mathbf{p}_B$ are found. We complete such remaining regions using the image completion technique [16] similarly to [6], [8]. In the current completed video, we first find the frame where the area of remaining regions is largest. We then apply image completion to the found frame and insert it back to the completed video. The voxels of the inserted frame belong to $\overline{\mathbf{H}}$ now, and we finally minimize Eq. 10 to update $\mathbf{V}$. We repeat this process while there exist remaining regions in $\mathbf{V}$.

3.4 Contrast Attenuation

The process described above causes artifacts in some cases. Fig. 4 shows an example. After removing the car from Fig. 4-a, we see artifacts of many undesirable edges on the road in Fig. 4-b. We explain about the situation where such an edge tends to be produced. Let $\mathbf{q}$ be the voxel position spatially neighboring to $\mathbf{p}$. Let $\mathbf{q}_{F'}$ and $\mathbf{q}_{B'}$ be the positions where the forward and backward trajectories through $\mathbf{q}$ first go outside the holes. Edges on the road shown in Fig. 4-b tend to be produced when F and F' are very different or B and B' are very different. To solve the problem, we attenuate the contrast between $\mathbf{p}$ and $\mathbf{q}$ in such a situation. We compute the desirable contrast between them as follows:

$$\begin{aligned} \mathbf{g}_{\mathbf{p},\mathbf{q}} &= (\mathbf{V}(\mathbf{p}) - \mathbf{V}(\mathbf{q})) \\ &\times \exp(-\beta(|F - F'| + |B - B'|)), \end{aligned} \tag{11}$$

where β is the user-specified parameter that controls the strength of attenuation. We then minimize the following energy function to update $\mathbf{V}$:

$$E^{contrast} = \sum_{\mathbf{p}\in\mathbf{P}} \sum_{\mathbf{q}\in\mathbf{N}_{\mathbf{p}}} |\mathbf{V}(\mathbf{p}) - \mathbf{V}(\mathbf{q}) - \mathbf{g}_{\mathbf{p},\mathbf{q}}|^2, \tag{12}$$

where $\mathbf{N}_{\mathbf{p}}$ represents $\{\mathbf{p} + (1, 0, 0), \mathbf{p} + (0, 1, 0)\}$. We solve this minimization per frame using a Poisson solver [17] and it takes 10 milliseconds for a 854×480 frame. The number of Jacobi iterations is 200 per frame. At each Jacobi iteration, we do not update $\mathbf{V}(\mathbf{p})$ at $\mathbf{p} \in \overline{\mathbf{H}}$. The result is shown in Fig. 4-c, where artifacts are successfully removed. We apply this contrast attenuation not always but when required.

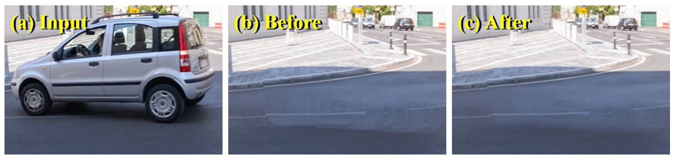

Fig. 4. After removing the car from (a), we see artifacts of many undesirable edges on the road in (b), which are successfully removed by our contrast attenuation in (c).

4 Results and Discussion

We applied our method to 33 videos. We downloaded pairs of videos of inputs and space-time holes from the project pages of the previous methods [4], [6], [7], [8]. All videos are from the DAVIS dataset [18]. The results and comparisons are shown in the supplementary video and material. The values of the open parameters (L and α) and the time required to produce each result are shown in the video of the supplementary material.

4.1 Computational Complexity

We used a desktop personal computer (PC) with an Intel(R) Core(TM) i7 4.0 GHz central processing unit (CPU), 32.0 GB of memory, and an NVIDIA GeForce GTX 1080 Ti graphics processing unit (GPU). We implemented our method so that almost all of the computations were performed by the GPU. The TV-L^1 optical flow estimation is so suitable for GPU implementation, which enables to develop a computationally efficient system. It took 4.1 seconds to complete the 'camel' video, which has a resolution of 854×480 pixels and a duration of 90 frames. We set L and α to 1 and 0.0. Table 1 shows a comparison between the computation time required using our method and those of previous methods. We did not measure the computation time for each previous method; rather, we referred to the values quoted in each paper.

The optical flow estimation is the most time-consuming task in our algorithm. For example, it took 88% of the total computation time to estimate the flow fields in the case of the 'camel' video. The computational complexity of the TV-L^1 optical flow algorithm is proportional to the number of iterations in it. We usually use 50 or 100 iterations at each pyramid level in the optical flow algorithm, but 1000 iterations are used to compute flow fields more carefully for

Method	Time
[6]	3 hours
[7]	50 minutes
[8]	75 seconds
Our method	4.1 seconds

TABLE 1
Computation times required to complete the 'camel' video.

'elephant', 'goat', and 'kite-surf', because they have complex motions and occlusions.

The computational complexity of our method is also affected by the number of pyramid levels L. In the case of the 'parkour' video, it took 5.4 seconds for $L = 1$ (Fig. 10-a), 6.7 seconds for $L = 2$ (Fig. 10-b), and 11 seconds for $L = 3$ (Fig. 10-c). Another open parameter α does not affect the computational complexity.

4.2 Comparison of Completed Videos

Our method estimates the color by minimizing E based on the L^2 data terms rather than the L^1 data terms. Technically, this is a simple extension, but it dramatically improves the quality of the completed videos. Fig. 5 shows typical examples. As the L^1 data terms bring temporally incoherent solutions, the colors of the voxels change suddenly, producing visible artifacts. The top row of Fig. 5 shows frames from the results for the 'parkour' video. The wall is broken in the result from [6] but is successfully reconstructed by our L^2 data terms. The bottom row of Fig. 5 shows frames from the results for the 'kite-surf' video. There are clearly visible seams between the blue water and the white splash in the result by [8], but these are successfully smoothed by our L^2 data terms. Such visible artifacts occur more often when the motions in the completed holes are more dynamic. Note that such artifacts have to be removed to produce satisfactory digital contents: digital artists often have to perform manual image editing frame by frame. The results based on our L^2 data terms can significantly reduce their burden.

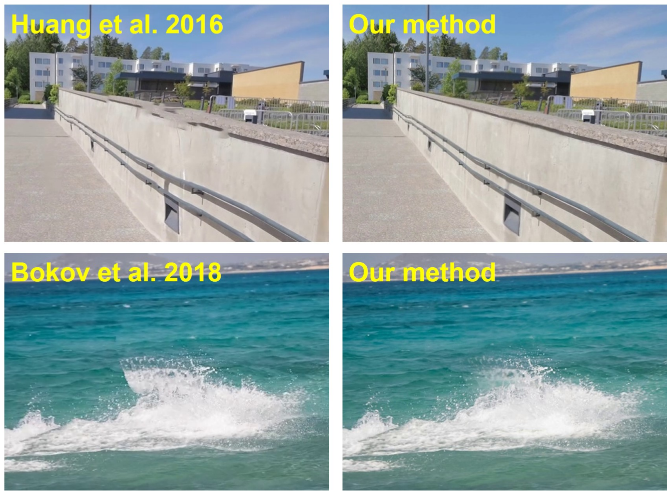

Fig. 5. Comparison between the results based on the L^1 data terms (left) and those based on the L^2 data terms (right). The L^1 data terms produce visible artifacts, which are successfully removed by our L^2 data terms.

4.3 Interactive Video Completion

Our method is computationally efficient, which means that we can provide an effective interactive tool. Figs. 6 and 7 show examples. The user interactively selects a frame and draws a mask to specify the objects that they want to remove. Once the drawing operation is complete, our system immediately interpolates between the user-drawn masks to create the space-time holes $\mathbf{H}$ and achieve video completion. It took 1 minute in total to remove the rollerblader from the 35-frame video with a resolution of 854×480 pixels in Fig. 6. It took 2.5 minutes in total to remove the bike from the 80-frame video with the same resolution in Fig. 7. Such a user interaction has never been demonstrated in the previous papers.

The reason why the user can remove the object from the video in such a short time is that the user can draw rough masks. As shown in Figs. 6 and 7, the drawn masks specify not only pixels of the target object but also pixels outside it. However, our method can efficiently completes such outside pixels by sampling corresponding pixels from the other frames.

Roughly drawn masks are often sufficient for video completion; hence, we computed the interpolation with a quarter of the original resolution for efficiency. We compute this interpolation by extending the image colorization method [19] to the spatio-temporal volume. Let $\mathbf{M}$ denote the interpolated mask that is the space-time volume with dimensions $W \times H \times T$ voxels. Each voxel of $\mathbf{M}$ has an intensity value from 0 to 1. We minimize the following energy function to compute $\mathbf{M}$:

$$\begin{aligned} E^{mask} &= \sum_{\mathbf{p}\in\mathbf{P}} \sum_{\mathbf{q}\in\mathbf{N_p}} \gamma(\mathbf{p},\mathbf{q})|\mathbf{M}(\mathbf{p}) - \mathbf{M}(\mathbf{q})|^2 \\ &+ 0.5\gamma(\mathbf{p},\mathbf{p}^f)|\mathbf{M}(\mathbf{p}) - \mathbf{M}(\mathbf{p}^f)|^2 \\ &+ 0.5\gamma(\mathbf{p},\mathbf{p}^b)|\mathbf{M}(\mathbf{p}) - \mathbf{M}(\mathbf{p}^b)|^2, \end{aligned} \quad (13)$$

where $\mathbf{N_p}$ represents $\{\mathbf{p} + (1,0,0), \mathbf{p} + (0,1,0)\}$, $\mathbf{p}^f = \mathbf{p} + \mathbf{F}_f(\mathbf{p})$, and $\mathbf{p}^b = \mathbf{p} + \mathbf{F}_b(\mathbf{p})$. We use the weighting function $\gamma(\mathbf{p},\mathbf{q}) = \exp(-\frac{|\mathbf{V}(\mathbf{p})-\mathbf{V}(\mathbf{q})|^2}{\sigma^2})$ to preserve the spatio-temporal edges of $\mathbf{V}$ during the interpolation. We solve this minimization using a Poisson solver. At each Jacobi iteration, we do not update $\mathbf{M}(\mathbf{p})$ in frames where the user has drawn a mask. Finally, we compute the set of voxel positions of the holes $\mathbf{H}$ as

$$\mathbf{H} = \{\mathbf{p}|\mathbf{M}(\mathbf{p}) > 0.5\}, \quad (14)$$

and apply our method to complete the video.

4.4 Manual Modification of Completed Videos

We now demonstrate that our approach based on the L^2 data terms is useful for both automatic video completion and manual modification of the results. There are visible artifacts in the results based on the L^1 data term (Fig. 1-a), which are clearly visible as seams in the $x - time$ slice. The results for the L^2 data term also contain artifacts, i.e., the flamingo has double legs (Fig. 1-b). To remove these artifacts, we modified frame 40 using the clone stamp tool in Adobe Photoshop. We inserted the modified frame into the input and reapplied our algorithm. Fig. 1-c and d show frame 28 of the results. The results based on the L^1 data term

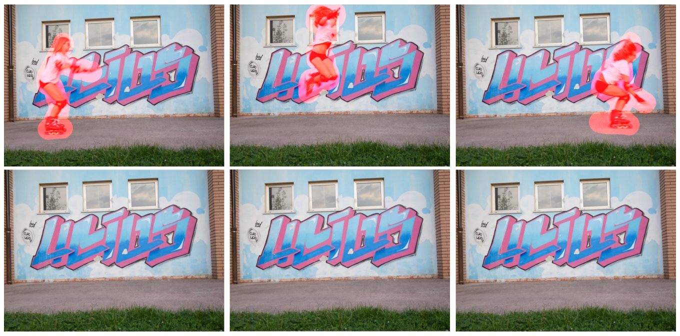

Fig. 6. It took 1 minute in total to remove the rollerblader from the 35-frame video with a resolution of 854×480 pixels. The top row shows the user-drawn masks and the buttom row shows the frames of the completed video.

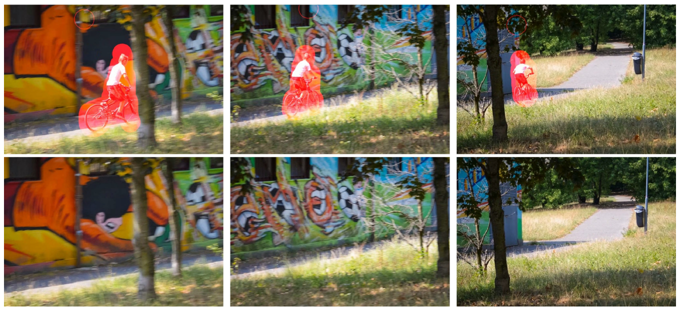

Fig. 7. It took 2.5 minutes in total to remove the bike from the 80-frame video with a resolution of 854×480 pixels. The top row shows the user-drawn masks and the buttom row shows the frames of the completed video.

still contain artifacts that are visible as temporal disconnections in the $x - time$ slice (Fig. 1-c). These artifacts are more noticeable in the supplementary video. On the other hand, the double legs artifacts were successfully removed by the method based on the L^2 data term (Fig. 1-d).

Our method is also useful to modify multiple frames to remove artifacts. For example, the fence in the completed 'parkour' video was broken by artifacts at frame 65 (Fig. 8-a). To fix the fence, we manually modified frames 62, 70, 73, and 75 and reapplied our algorithm. While the fence in the result based on the L^1 data term is still broken (Fig. 8-b), that based on the L^2 data term is better fixed (Fig. 8-c).

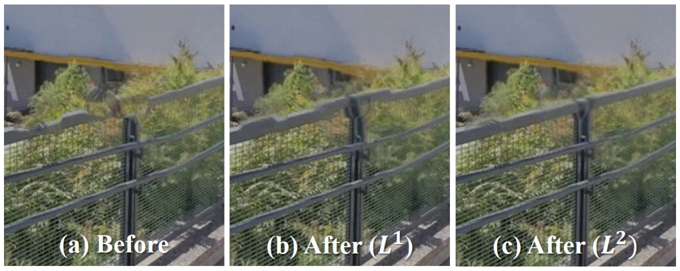

Fig. 8. Our method, based on the L^2 data term, better fixes the broken fence after modification of multiple frames.

4.5 Manual Modification on Remaining Regions

Our method automatically completes the remaining regions by applying an image completion technique (Sec. 3.3). However, this technique often fails to produce an aesthetically pleasing result. Fig. 9 shows examples of such failures in the remaining regions. The rail in the 'train' video is broken (Fig. 9-a). The synthesized textures around the fallen tree look unnatural in the 'rhino' video (Fig. 9-c). We then selected a single frame from the completed video and manually modified it using the clone stamp tool. We inserted the modified frame into the input video, then reapplied our algorithm to produce better results, such as Fig. 9-b and 9-d. Since our method is fast, the user can easily repeat such manual modifications.

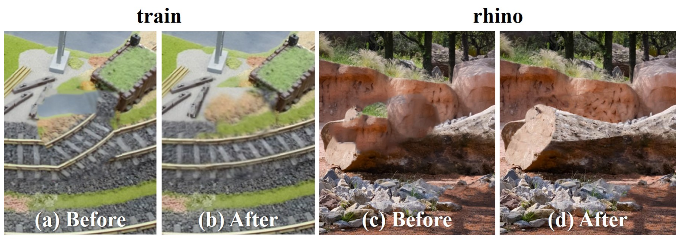

Fig. 9. Manual modifications on automatically completed remaining regions (a and c) produce better results (b and d).

4.6 Open Parameters

Our iterative algorithm has two open parameters: the number of pyramid levels L and α that affects weights for the data terms in Eq. 1. The larger L and α bring temporally smoother color transitions. Fig. 10-a shows the result based on $L = 1$, where many ghosting artifacts are visible. This is because the images sampled by tracking $\mathbf{F}_f$ and $\mathbf{F}_b$, which correspond to $\mathbf{V}(\mathbf{p}_F)$ and $\mathbf{V}(\mathbf{p}_B)$, look so different. On the other hand, when we set L to 3, $\mathbf{F}_f$ and $\mathbf{F}_b$ are optimized so that $\mathbf{V}(\mathbf{p}_F)$ and $\mathbf{V}(\mathbf{p}_B)$ produce similar images, which results in more natural-looking completed video (Fig. 10-c).

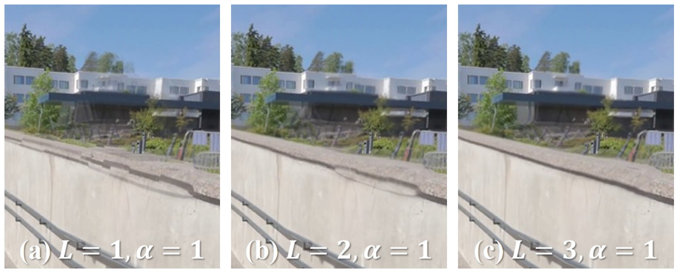

Fig. 10. The effects of the number of pyramid levels. Ghosting artifacts are visible in the result on $L = 1$ (a), which are removed in the result on $L = 3$ (c).

4.7 Contrast Attenuation

We applied contrast attenuation (Sec. 3.4) to 'bmx-bumps', 'breakdance-flare', 'car-shadow', and 'motorbike'. We set β to 1 for the 'breakdance-flare' video and 10 for the other videos. It successfully removes artifacts in some cases as shown in Fig. 4. On the other hand, since it does not take temporal coherence into account but processes each frame independently, flickering artifacts often appear, which are clearly visible in the completed 'breakdance-flare' video. To avoid such artifacts, we apply contrast attenuation only to sparsely selected frames, insert them back to the input video, and reapply our algorithm. The modified version of the completed 'rhino' video was produced in such a way that, after the manual modification (Sec. 4.5), we applied the contrast attenuation to 0-th, 20-th, 60-th, 75-th, and 89-th frames. We then inserted them back into the corresponding frames of the original video and our video completion algorithm was reapplied.

4.8 Subjective Evaluations

We subjectively evaluated the visual quality of the completed videos. This evaluation clearly indicates that our L^2 scheme proves to be superior to the L^1 scheme of all the other previous methods. The researchers seem to believe that the L^1 scheme is the best. However, this evaluation reveals that this belief is not true and our L^2 scheme is actually better.

We used 24 students (19 males and 5 females) as participants, all of whom major in computer science or engineering and are not familiar with video completion. First, we evaluated our method (without any manual modification) and the three methods presented in [4], [6], [8], as the experimental results obtained using these methods were produced using the space-time holes on the project page for [6]. We used 25 input videos; hence, there were four completed videos for each input video, i.e., each subject evaluates 100 ($= 25 \times 4$) videos. We did not include 'dance-twirl' in this subjective evaluation, since the completed videos had a different duration. We dilated the space-time holes for the 'elephant' result of our method to avoid the influence of sand and dust.

For each input video, we asked the subject to watch both the input and completed videos. We then asked the subject to sort the completed videos based on quality, i.e., whether the completed video looks natural and is free from visible artifacts. We asked the subjects to repeat this task for all 25 input videos, which were presented in a random order. We allowed each subject to freely change the size of the video player and watch each video as many times as they liked. Thus, we obtained 600 ($= 24$ subjects $\times$ 25 input videos) rankings. Fig. 11 shows the number of being ranked 1st for each input video.

The videos completed by our method were evaluated as the best, i.e., they were ranked 1st most often, for 21 of the 25 input videos. The input videos that were ranked higher when completed using other methods were the bottom four in Fig. 11, i.e., 'motorbike', 'breakdance', 'rollerblade', and 'breakdance-flare'. Our results of 'motorbike' and 'breakdance' were 2nd ranked, but they are almost comparable to those of the other methods.

We performed manual modifications on the results that were unsatisfactory, and asked subjects to watch these modified versions. Fig. 12 shows the result. Our manual modifications for 'flamingo' (Fig. 1) and 'parkour' (Fig. 8) successfully improve the quality of the completed videos. The reason for our poorer performance on the 'elephant', 'rhino', and 'train' videos in Fig. 11 was that the quality of our image completion on the remaining regions (Sec. 3.3) was poor (see Fig. 9-a and 9-c). We then modified each result

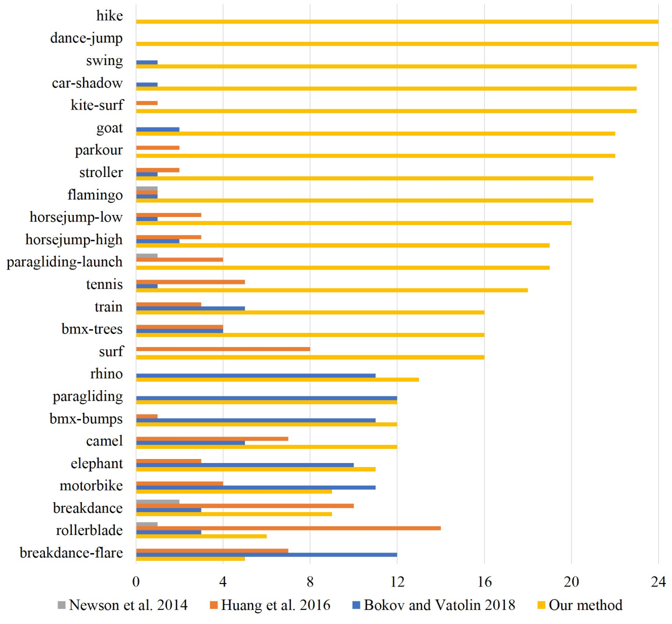

Fig. 11. Results of the subjective evaluation. Each bar represents the number of subjects who ranked the corresponding method 1st.

by manually modifying a specific frame and reapplied our algorithm. These modified videos (as in Fig. 9-b and 9-d) were ranked highest, i.e., they were ranked 1st most often. Our result for the 'rollerblade' video was not as good as those obtained using the other methods. We then asked each participant to evaluate the interactive video completion result, in which the rollerblader's shadows were better removed (Fig. 6), and it was ranked highest.

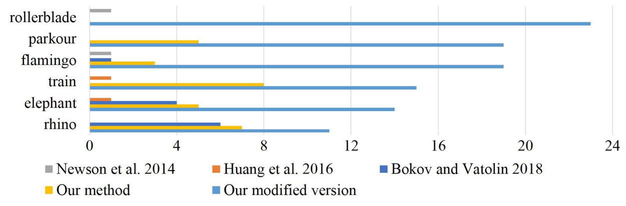

Fig. 12. Results of the subjective evaluation for the manually modified versions. Each bar represents the number of subjects who ranked the corresponding method 1st.

We also evaluated our method and the method presented in [7] independent of the other three methods, as Le et al. used different space-time holes. We used 32 input videos, and there were two completed videos for each input video, which we asked the subjects to rank.

The videos completed by our method were ranked 1st for 25 of the 32 videos. The remaining seven videos were 'blackswan', 'breakdance', 'drift-chicane', 'kite-walk', 'mallard-water', 'motorbike', and 'rollerblade'. As Le et al.'s method uses a patch-based approach, it is good at synthesizing dynamic textures, e.g., repetitive motion of clapping hands, smoke, and water surface, which our method fails to complete. Our method did not remove the shadows for the 'rollerblade' as efficiently as Le et al.'s method. We then asked each participant to evaluate the interactive video completion result again (Fig. 6), and it was ranked highest.

5 Conclusion and Future Work

We have proposed a practical video completion method. Our method is computationally efficient that allows the user to perform video completion by interactively drawing masks. Our method minimizes the energy function based on the L^2 data terms to estimate temporally coherent color transitions, which not only produces natural-looking results but also is useful for manual modifications on single or multiple selected frames. The subjective evaluation results also illustrate very well that our method has superiority over previous video completion approaches.

However, there still exist lots of avenues for making our method a production-ready tool for video completion.

1) Since the success of video completion relies on the quality of masks to specify space-time holes, we are also interested in extraction of good masks from the input video like [20]. To address these issues, we are interested in exploiting more semantic information, e.g., that would be provided by deep learning based approaches for image and video understanding.
2) Theoretically, our method works on high res videos, such as 4K or 8K, but it often becomes inefficient due to out of memory. Our method consumes a large amount of memory, especially to keep the forward and backward flow fields at the same time. For example, we require 3.2 GB of memory to keep them for a 100-frame video at 2K resolution. This will be problematic when completing a longer, higher-resolutional video. We want to investigate the way to use memory more efficiently.
3) We want to fix our failure cases of repetitive motions, fluid motions, etc. The patch-based approach is good at synthesizing such dynamic textures, but the methods based on it are usually computationally expensive. We want to explore an efficient patch-based approach that enables the user to interactive edits of completed videos.
4) The current contrast attenuation is not easy-to-use for two reasons: 1) it does not take temporal coherence into account, which often causes flickering artifacts; 2) as the result, this is currently an optional tool, i.e., the user has to decide whether it should be applied or not. We want to improve the algorithm and propose an easier-to-use tool.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful and constructive comments. Many thanks also go to So Hashimoto, and Ayumi Kimura for discussions and encouragements. This work was partially supported by JSPS KAKENHI Grant Numbers JP15H05924 and JP18K11352.

References

[1] Y. Wexler, E. Shechtman, and M. Irani, "Space-time completion of video," *IEEE TPAMI*, vol. 29, no. 3, 2007.
[2] M. Granados, J. Tompkin, K. Kim, O. Grau, J. Kautz, and C. Theobalt, "How not to be seen-object removal from videos of crowded scenes," *Computer Graph. Forum*, vol. 31, no. 2pt1, pp. 219–228, 2012.
[3] M. Granados, K. I. Kim, J. Tompkin, J. Kautz, and C. Theobalt, "Background inpainting for videos with dynamic objects and a free-moving camera," in *Proc. ECCV 2012*, 2012, pp. 682–695.
[4] A. Newson, A. Almansa, M. Fradet, Y. Gousseau, and P. Pérez, "Video inpainting of complex scenes," *SIAM Journal on Imaging Sciences*, vol. 7, no. 4, 2014.
[5] M. Roxas, T. Shiratori, and K. Ikeuchi, "Video completion via spatio-temporally consistent motion inpainting," *IPSJ Transactions on Computer Vision and Applications*, vol. 6, pp. 98–102, 2014.
[6] J.-B. Huang, S. B. Kang, N. Ahuja, and J. Kopf, "Temporally coherent completion of dynamic video," in *Proc. SIGGRAPH Asia 2016*, 2016, pp. 196:1–196:11.
[7] T. T. Le, A. Almansa, Y. Gousseau, and S. Masnou, "Motion-consistent video inpainting," in *Proc. IEEE ICIP 2017*, 2017, pp. 2094–2098.
[8] A. Bokov and D. Vatolin, "100+ times faster video completion by optical-flow-guided variational refinement," in *Proc. IEEE ICIP 2018*, 2018, pp. 2122–2126.
[9] R. Murase, Y. Zhang, and T. Okatani, "Video-rate video inpainting," in *Proc. IEEE WACV 2019*, 2019, pp. 1553–1561.
[10] T. Shiratori, Y. Matsushita, S. B. Kang, and X. Tang, "Video completion by motion field transfer," in *Proc. CVPR 2006*, 2006, pp. 411–418.
[11] M. Strobel, J. Diebold, and D. Cremers, "Flow and color inpainting for video completion," in *Proc. CVPR 2014*, 2014, pp. 293–304.
[12] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool, "Fast Optical Flow using Dense Inverse Search," in *Proc. ECCV 2016*, 2016.
[13] A. Nandoriya, M. Elgharib, C. Kim, M. Hefeeda, and W. Matusik, "Video reflection removal through spatio-temporal optimization," in *IEEE ICCV*, 2017, pp. 2411–2419.
[14] R. Sadek, G. Facciolo, P. Arias, and V. Caselles, "A variational model for gradient-based video editing," *International Journal of Computer Vision*, vol. 103, pp. 127–162, 2013.
[15] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l1 optical flow," in *Proc. DAGM*, 2007, pp. 214–223.
[16] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," in *Proc. SIGGRAPH 2009*, 2009.
[17] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," in *Proc. SIGGRAPH 2003*, 2003, pp. 313–318.
[18] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *Proc. CVPR 2016*, 2016.
[19] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," in *Proc. SIGGRAPH 2004*, 2004, pp. 689–694.
[20] T. T. Le, A. Almansa, Y. Gousseau, and S. Masnou, "Removing objects from videos with a few strokes," in *Proc. SIGGRAPH Asia 2018 Technical Briefs*, 2018, pp. 22:1–22:4.