# Temporally Coherent Completion of Dynamic Video

Jia-Bin Huang
Virginia Tech*

Sing Bing Kang
Microsoft Research

Narendra Ahuja
University of Illinois, Urbana-Champaign

Johannes Kopf
Facebook


**Figure 1:** *We present a fully automatic algorithm for plausibly completing missing regions in videos of dynamic scenes and captured with a moving camera in a temporally coherent manner. The top row shows sample frames from a hand-held video of a dancer. Both the foreground and background, as well as the camera are moving throughout the video. The red masks are user-selected regions to be removed. The bottom row shows our automatic completion result.*

**Keywords:** Video completion, patch-based synthesis

**Concepts:** •**Computing methodologies** → **Computational photography; Image manipulation;**

## Abstract

We present an automatic video completion algorithm that synthesizes missing regions in videos in a temporally coherent fashion. Our algorithm can handle dynamic scenes captured using a moving camera. State-of-the-art approaches have difficulties handling such videos because viewpoint changes cause image-space motion vectors in the missing and known regions to be inconsistent. We address this problem by jointly estimating optical flow and color in the missing regions. Using pixel-wise forward/backward flow fields enables us to synthesize temporally coherent colors. We formulate the problem as a non-parametric patch-based optimization. We demonstrate our technique on numerous challenging videos.

## 1 Introduction

Video completion methods are designed to fill user-specified spatio-temporal holes with plausible content using remaining parts of the video. An effective video completion method has many practical applications in video post-production, such as unwanted object removal, full-frame video stabilization (as a byproduct), logo or watermark removal in broadcast videos, and restoration of damaged vintage films.

Much progress has been made on automatic single-image completion, to a point of where commercial solutions are now available.[1] However, automatic video completion algorithms have fared less well. This is due to the additional time dimension which introduces major challenges: (1) viewpoint changes cause non-trivial appearance changes in image-space; (2) the synthesized content needs to be temporally coherent; (3) there is exponentially increased computational complexity due to the larger number of missing pixels.

Many state-of-the-art video completion algorithms synthesize the missing (target) regions by sampling spatio-temporal patches from the known (source) regions [Wexler et al. 2007; Newson et al. 2014] or by solving spatio-temporal shift-maps using graph cuts [Granados et al. 2012b]. While good results have been shown, these approaches have two major limitations.

One limitation for translation-based sampling is the degradation caused by source and target regions having inconsistent color, texture, or motion, which may be caused by viewpoint changes in hand-held videos or non-periodic object motion. The inconsistency problems can be partially alleviated by warping spatio-temporal object samples [Jia et al. 2006], stabilizing the input video [Newson et al. 2014], or compensating geometric distortion through projective transformation [Granados et al. 2012a]. However, these approaches assume either a clean foreground and background layer separation, simple parametric global motion, or static background and therefore have difficulties in handling general situations.

Another limitation is that motion is not explicitly reconstructed even though motion-based features are used as part of the similarity metric. As shown in our experiments, video completion algorithms without explicit motion reconstruction often yield results

*Part of this work was done while Jia-Bin was a research assistant at University of Illinois, Urbana-Champaign.


SA '16 Technical Papers,, December 05-08, 2016, , Macao
ISBN: 978-1-4503-4514-9/16/12$15.00
DOI: http://dx.doi.org/10.1145/2980179.2982398

[1] See, for example, http://www.adobe.com/technology/projects/content-aware-fill.html

**ACM Reference Format**
Huang, J., Kang, S., Ahuja, N., Kopf, J. 2016. Temporally Coherent Completion of Dynamic Video. ACM Trans. Graph. 35, 6, Article 196 (November 2016), 11 pages. DOI = 10.1145/2980179.2982398 http://doi.acm.org/10.1145/2980179.2982398.

**Table 1:** *Comparisons with state-of-the-art video completion algorithms. The cells highlighted in red indicate limitations of an algorithm. The optimization techniques used in [Newson et al. 2014] and our approach that alternate between patch search and patch voting steps can be viewed as a "Hard EM" algorithm for estimating maximum-likelihood solution [Wexler et al. 2007]. The visibility assumption refers to each missing pixel needing to be visible in at least one frame.*

| Method | [Granados et al. 2012a] | [Granados et al. 2012b] | [Newson et al. 2014] | [Strobel et al. 2014] | Ours |
|---|---|---|---|---|---|
| Synthesis unit | Spatial | Spatiotemporal | Spatiotemporal | Spatial | Spatial |
| Optimization | Graph-cut | Graph-cut | Hard-EM | Greedy | Hard-EM |
| Dynamic background | No | Yes | Yes | No | Yes |
| Visibility assumption | Yes | No | No | No | No |
| Flow estimation | No | No | No | Yes | Yes |
| Warping | Homography | N/A | Affine | N/A | Dense flow field |
| Temporal consistency | No | No | No | Yes | Yes |

that are plausible when viewed as separate images, but are not temporally coherent. Several recent approaches address the temporal consistency problem by explicitly synthesizing flow field in the target regions [Strobel et al. 2014; Roxas et al. 2014]. However, these approaches smoothly interpolate the flow field with diffusion-based techniques, and are less effective in synthesizing dynamic backgrounds.

In this paper, we present a new video completion algorithm that does not make any simplifying assumptions about the video content; it works on casual hand-held videos with moving camera, dynamic content, lighting and color variations, and missing pixels that are not seen in any of the known regions. Our method can fill missing regions in such videos with plausible content in a temporally coherent manner.

Our method jointly estimates appearance (color) and dense flow fields in the missing regions. The key idea is that the reconstructed pixel-wise forward and backward flow fields allow us to explicitly promote temporal coherence. Since our flow field is non-parametric, it can handle general motion parallax and dynamic scenes. This is in comparison with existing video completion algorithms that use parametric motion to mostly compensate for camera motion (e.g., affine [Newson et al. 2014] or homography [Granados et al. 2012a]). We formulate video completion as non-parametric patch-based optimization. The combination of non-parametric spatial patch-based optimization and dense flow field estimation facilitate synthesizing colors that are spatially coherent (i.e., locally appear similar to the known regions everywhere) in each frame while maintaining temporal coherence (i.e., with small flow warp errors) across frames.

We have tested our algorithm on numerous challenging videos that span a wide variety of situations, including hand-held and stationary cameras, static and dynamic backgrounds, and multiple moving objects. We show representative still frames from the videos throughout the paper, and full video results can be found in the supplementary material.

## 2 Related Work

Image and video completion are well-explored topics; surveys by Guillemot and Le Meur [2014] and Ilan and Shamir [2015] provide a more comprehensive review. Here we limit our discussion to state-of-the-art or representative approaches. Table 1 shows a feature-by-feature comparison with representative techniques.

**Patch-based synthesis** These methods fill missing regions by sampling non-local spatio-temporal patches (e.g., $5 \times 5 \times 5$) from the known input. Efros and Leung [1999] first introduced non-parametric sampling techniques for texture synthesis. The approach was later extended and applied to image completion [Criminisi et al. 2004; Drori et al. 2003] and video completion [Patwardhan et al. 2005; Patwardhan et al. 2007]. However, for video completion, these techniques either assume static cameras [Patwardhan et al. 2005] or constrained camera motion [Patwardhan et al. 2007] so that the foreground and background layers can be easily separated and independently filled. Furthermore, the greedy patch filling process inevitably propagates errors at early steps to the subsequent steps, yielding globally inconsistent results.

To address the global inconsistency issue, patch-based completion algorithms have been cast as a global optimization problem that is minimized by alternating between patch search and reconstruction steps [Wexler et al. 2007; Kwatra et al. 2005]. Newson et al. [2014] recently extend the patch-based optimization approach by incorporating texture features, compensate dominant camera motion with global affine transformation, and use a spatio-temporal version of PatchMatch [Barnes et al. 2009] for fast approximated nearest neighbor search.

Our algorithm builds upon the non-parametric optimization framework with three major differentiators. First, we fill the hole by sampling *spatial* patches rather than *spatio-temporal* patches. While spatio-temporal patches provide a simple way for encoding local appearance *and* motion, the assumption that spatio-temporal blocks appear repeatedly through time is typically not valid under the hand-held camera condition and non-repetitive scene motion. Spatial patches, on the other hand, have no such restriction and thus are applicable to general scenarios. Second, our approach explicitly estimates dense forward and backward flow fields. The estimated dense flow fields allow us to explicitly enforce temporal coherence of the synthesized contents as well as propagate known content into the unknown regions. Third, similar to Darabi et al. [2012], we augment the patch search space to account for texture and structural inconsistency between the source and the target regions. In addition to random search, we use the local flow vectors to predict and propagate transformation parameters from frame to frame.

**Segment-based synthesis** Such methods pose completion as a labeling problem by solving a correspondence map (or a shift map) between the source and target pixels. They typically apply graph cuts to find optimal seams and thus do not need to specify patch sizes. Kwatra et al. [2003] propose a texture synthesis algorithm that iteratively selects a shifted version of an input texture and uses graph cuts to find minimally noticeable seams between the original and shifted textures. Moving beyond texture synthesis, Pritch et al. [2009] generalize segment-based approaches to several other image editing tasks, including image reshuffling, retargeting and completion using a global multi-label graph cut based optimization. Granados et al. [2012b] extend this method to video completion and introduce an interactive interface for users to help

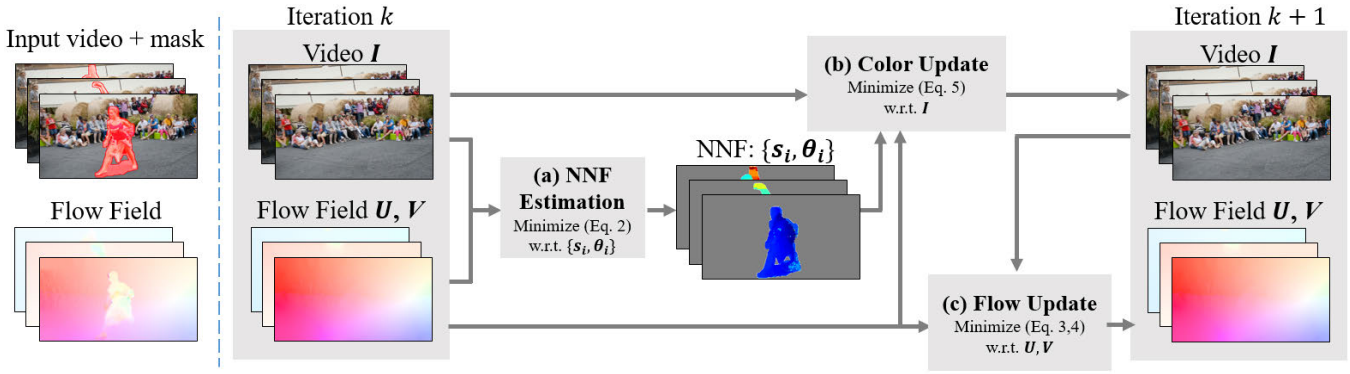



**Figure 2:** *Algorithm pipeline. Given the input video and user-selected mask, we start with computing the flow fields. After initialization at the coarsest scale (Section 4.4), in each scale our algorithm iterates through three steps (Section 4.3): (a) nearest neighbor field estimation: minimize the color spatial cost by finding dense approximate nearest neighbor source patches for all target patches; (b) color update: minimize the color spatial and color temporal cost so that the synthesized colors are both spatially and temporally coherent; and (c) flow update: refine the forward and backward flow fields. We then upsample the solution of the nearest neighbor field and flow fields to the next finer level. The color at the finer level is estimated by spatial patch voting (using the upsampled nearest neighbor field).*

constrain the search space.

The major limitation of segment-based methods is that the synthesized content has to be copied from the unoccluded regions "as is" in the known input. Therefore, these methods in general cannot handle objects that undergo appearance changes (e.g., scale variations when an object moves toward or away from the camera) as well as video captured with a hand-held camera. Granados et al. [2012a] show that the appearance variations between the source and target regions can be compensated by homographies. However, this approach assumes manual labeling of foreground regions, piecewise planar and static background, and visibility of the occluded region (i.e., the occluded region must be visible in some other frames). In contrast, our method offers greater flexibility for handling appearance variations and does not have above mentioned limitations.

**Flow-based synthesis** To address the temporal consistency problem, techniques have been developed to the fill motion field in missing regions, e.g., through a greedy selection of spatio-temporal patches of local motion [Shiratori et al. 2006], per frame diffusion [Strobel et al. 2014; Matsushita et al. 2006], or iterative optimization [Roxas et al. 2014] with propagated colors from the known boundary.

Our method differs in two ways. First, instead of treating flow estimation as an independent step from color estimation [Shiratori et al. 2006; Strobel et al. 2014; Matsushita et al. 2006], our approach iteratively computes and refines the dense flow field from the synthesized colors and vice versa. Second, filling the colors only through propagating from the known boundary (e.g., [Shiratori et al. 2006; Roxas et al. 2014]) inevitably generates blurry results due to successive averaging of colors and thus are applicable only for holes with very narrow temporal span. In contrast, we synthesize color by patch-based optimization and use the flow to enforce the temporal consistency. As a result, our approach can handle holes with arbitrary temporal span.

**Flow-based video processing** Our work is also related to several flow-based video synthesis and editing tasks. Flow representation provides a direct way to enforce temporal consistency for texture synthesis [Kwatra et al. 2005], fluid animation [Jamriška et al. 2015], video denoising [Liu and Freeman 2010], video editing [Bhat et al. 2004], morphing [Shechtman et al. 2010], looping video generation [Sevilla-Lara et al. 2015], high dynamic range video reconstruction [Kalantari et al. 2013]. The work most related to our work is that of Xue et al. [2015], which decomposes a set of images into a clean background and occlusion/reflection layers. Unlike [Xue et al. 2015], our approach does not require every pixel in the occluding region to be visible in at least one frame. That is, we are able to hallucinate plausible contents for the unknown regions that are *not* visible in the entire image sequence.

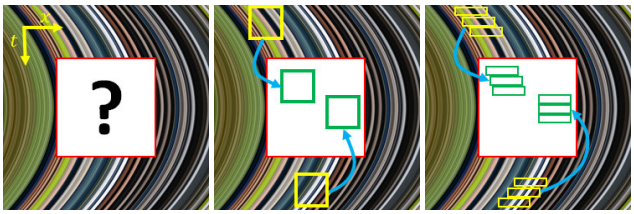
**Algorithm 1:** Proposed video completion algorithm.

**Input** : Video $\boldsymbol{I}$, user-specified mask $\overline{\Omega}$
**Output**: Completed video $\boldsymbol{I}$

```
Compute forward/backward flow fields U,V in Ω
Initialization: filling hole Ω̄ in I,U,V at coarsest scale (Sec. 4.4)
for scale s from 1 to n_s do
    for iteration k from 1 to K_s do
        (a) NNF estimation:
        Minimize Eq. 2 w.r.t. {s_i, θ_i}, with I,U,V fixed.
        (b) Color update:
        Minimize Eq. 5 w.r.t. I, with U,V,{s_i, θ_i} fixed.
        (c) Flow update:
        Minimize Eqs. 3 and 4 w.r.t. U,V, with I,{s_i, θ_i} fixed.
    end
    Upsample U,V using bicubic interpolation.
    Upsample {s_i, θ} using nearest-neighbor interpolation.
end
```

## 3 Overview

In our video completion algorithm, we jointly estimate the unknown color values and local motion in the target regions. We start with computing forward and backward optical flow in the *known* region for all adjacent frames using a two-frame optical flow algorithm [Liu 2009]. We implement our color synthesis algorithm similar to the non-parametric patch-based optimization algorithms of Wexler et al. [2007] and Kwatra et al. [2005]. In each iteration, we loop over three main steps: (1) patch search, (2) color update




(a) Input and hole Spatiotemporal Spatial

**Figure 3:** *Limitation of using spatio-temporal patches/segments. We use a frame from sequence* DANCE-TWIRL *and synthetically generate translational motion along the x-axis. (a) A spatio-temporal x-t slice of the sequence with mask overlay. (b) Using spatiotemporal patches (2D patches here) is not able to properly fill the missing region because the motion between the source (yellow) and target (green) regions are not consistent. (c) Using spatial patches (1D slices here), on the other hand, offers greater flexibility by adapting to the local flow.*

and (3) forward/backward flow update. We summarize our method in Algorithm 1 and illustrate the pipeline in Figure 2.

**Patch search** In this step, for all overlapping patches in the target regions, we search their corresponding patches in the source region with most similar local appearances. Our algorithm differs from existing non-parametric patch sampling methods in two major aspects. First, in contrast to methods using spatio-temporal patches (e.g., $5\times5\times5$ pixels) for synthesis, we use *spatial-only* patches ($5\times5$ patch in our experiments) as synthesis units. We enforce temporal consistency by augmenting the patch matching cost with color consistency along the flow vectors. Such representation allows us to handle complex flow fields caused by arbitrary camera motion and scenes with depth variations. For example, in a video captured by a hand-held camera, spatial patches with the exact same appearance may be embedded in significantly different spatio-temporal patches because the spatially-varying motion field caused by the moving camera. Therefore, working with spatio-temporal patches would have fundamental limitations in exploiting the non-local repetition of patches for video completion. In contrast, working with spatial patches does not suffer from such cases (Figure 3). Second, in contrast to copying exactly the local color patches from the source to the target region, we augment the patch search space to accommodate geometric variations. Specifically, we search additional geometric transformation (scale and rotation) of patches. Similar to the single-image case [Darabi et al. 2012], the additional patch transformation offers greater flexibility in synthesizing visually plausible contents even with limited geometric variations in the source regions.

**Color update** In the conventional patch-based optimization framework, the reconstruction step involves "patch voting," where the pixel color at an unknown pixel is estimated by averaging the colors of the corresponding nearest neighbors from all overlapping patches [Wexler et al. 2007]. This encourages spatial coherence, i.e., reconstructed patches look similar to some place in the known region. We extend this step to incorporate also diffused colors from the nearest *transitive* temporal neighbors in the known region (i.e., found by walking along the flow vectors until a known pixel is reached). We use forward/backward flow consistency to identify areas of unreliable flow, e.g., in occluded or dis-occluded regions.

**Forward and backward flow update** After estimating color in the target region, we update and refine the forward and backward flow fields. This step resembles the two-frame optical flow computation in the known region. Here, we fix the flow in the source regions and only update the flow in the target regions, resulting in an optical flow estimation algorithm with spatio-temporal hole boundary constraints.

## 4 Completion as Optimization

In this section, we provide details for our approach. We start by introducing the problem formulation and objective function. Next, we describe our optimization procedure for joint color and flow estimation.

### 4.1 Problem formulation

Let $\boldsymbol{I}$ be the input video of height $H$, width $W$ and number of frames $L$, and $\boldsymbol{U},\boldsymbol{V}$ the forward and backward flow fields, respectively. The forward flow at position $(x,y,f)$ is given by $\boldsymbol{U}(x,y,f) = (\mathrm{d}x,\mathrm{d}y,+1)$, indicating the flow vector $(\mathrm{d}x,\mathrm{d}y)$ from a point located at $(x,y,f)$ to a point $(x+\mathrm{d}x,\ y+\mathrm{d}y,\ f+1)$ in the video $\boldsymbol{I}$. Similarly, the backward flow at $(x,y,f)$ is $\boldsymbol{V}(x,y,f) = (\mathrm{d}x,\mathrm{d}y,-1)$. We denote the set of unknown pixels by $\overline{\Omega}$ (i.e., the user-specified spatio-temporal regions) and the set of known pixels by $\Omega$.

We define $\boldsymbol{t}_i = (t_i^x, t_i^y, t_i^f) \in \overline{\Omega}$ as the $i^{th}$ target pixel position, where $(t_i^x, t_i^y)$ is the 2D spatial position and $t_i^f$ is the frame index. Our goal is to estimate the unknown color values $\boldsymbol{I}(\boldsymbol{t}_i)$ for all target pixels, as well as the forward/backward flow vectors $\boldsymbol{U}(\boldsymbol{t}_i), \boldsymbol{V}(\boldsymbol{t}_i)$. We estimate the colors through non-parametric patch-based optimization. Specifically, for the $i^{th}$ target (unknown) pixel, we seek source (known) pixel position $\boldsymbol{s}_i = (s_i^x, s_i^y, s_i^f) \in \Omega$ and 2D patch geometric transformation $\theta_i \in \mathbb{R}^2$ (patch scaling and rotation) to minimize spatial reconstruction errors. Temporal consistency of synthesized colors $\boldsymbol{I}(\boldsymbol{t}_i)$ is achieved by enforcing color consistency along forward and backward flow fields $\boldsymbol{U},\boldsymbol{V}$. We describe the objective function and iterative optimization steps in the following subsections.

### 4.2 Objective function

We solve the following problem:

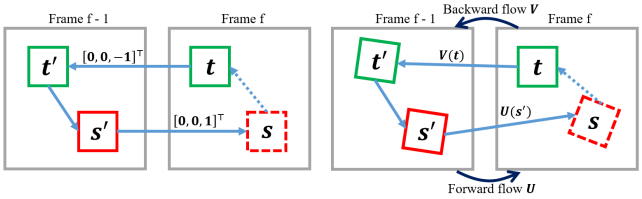
$$\underset{\boldsymbol{I},\boldsymbol{U},\boldsymbol{V},\{\boldsymbol{s}_i,\theta_i\}}{\operatorname{argmin}} \ \boldsymbol{E}_{\textit{color-spatial}} + \boldsymbol{E}_{\textit{color-temporal}} + \boldsymbol{E}_{\textit{flow-spatial}}, \qquad (1)$$

where $\boldsymbol{E}_{\textit{color-spatial}}$ penalizes *spatial* patch-based reconstruction errors of colors in the target regions, $\boldsymbol{E}_{\textit{color-temporal}}$ penalizes *temporal* inconsistency of the synthesized colors, and $\boldsymbol{E}_{\textit{flow-spatial}}$ is a spatial regularization term for the forward and backward flow fields.

**Spatial color cost** This cost encourages local neighborhoods $P_i$ in the target region to appear similar to local neighborhoods $Q_i$ in the source region. It also encourages spatial coherence, since we consider overlapping target patches. The target patch $P_i$ is an axis-aligned $5\times5$ spatial color patch, centered around $\boldsymbol{t}_i$. The source patch $Q_i$ is also a $5\times5$ patch that results from the scale and rotation transformation specified by $\theta_i$ centered at $\boldsymbol{s}_i$. The spatial color cost is the sum of square losses for all overlapping patches in $\overline{\Omega}$ and their correspondences:

$$\boldsymbol{E}_{\textit{color-spatial}}(\boldsymbol{I},\{\boldsymbol{s}_i,\theta_i\}) = \sum_{i\in\overline{\Omega}} \|P_i - Q_i\|_2^2 . \qquad (2)$$

We use a Gaussian falloff function with $\sigma = 1$ to give higher weights to pixels closer to the center of the neighborhood.




(a) PatchMatch propagation (b) Flow-guided propagation

**Figure 4:** *Flow-guided temporal propagation. (a) The direct extension of the PatchMatch algorithm [2009] to 3D [Newson et al. 2014]. Similar to the spatial propagation case in PatchMatch, candidate patches are propagated along the temporal axis. (b) The proposed flow-guided temporal propagation relaxes the constraints of axis-aligned propagation and uses local forward and backward flow vectors for accurate prediction of the candidate source patch position and transformation.*

**Temporal color cost** This cost encourages temporal color consistency between adjacent frames along the forward and backward flow vectors. We use a term that is commonly employed in optical flow algorithms:

$$\boldsymbol{E}_{\textit{color-temporal}}(\boldsymbol{I},\boldsymbol{U},\boldsymbol{V}) = \sum_{i\in\Omega} \alpha\phi\Big(\big\|\boldsymbol{I}(\boldsymbol{t}_i) - \boldsymbol{I}\big(\boldsymbol{t}_i + \boldsymbol{U}(\boldsymbol{t}_i)\big)\big\|_2^2\Big) + \alpha\phi\Big(\big\|\boldsymbol{I}(\boldsymbol{t}_i) - \boldsymbol{I}\big(\boldsymbol{t}_i + \boldsymbol{V}(\boldsymbol{t}_i)\big)\big\|_2^2\Big), \tag{3}$$

where the Charbonnier penalty $\phi(x^2) = \sqrt{x^2+\varepsilon}$ with a small constant $\varepsilon$ is a robust function (a differentiable variant of $\ell_1$-norm), and $\alpha = 0.5$ is the weight coefficient for this cost.[2]

While this cost in Eq. 3 directly encourages consistency for *all* temporally neighboring pixels, in practice we find it more effective to minimize a similar cost that couples the current pixel and its transitive forward/backward neighbor, as described in Section 4.3.

**Spatial flow cost** To promote piecewise smooth flow fields, we introduce spatial regularization:

$$\boldsymbol{E}_{\textit{flow-spatial}}(\boldsymbol{U},\boldsymbol{V}) = \beta \sum_{i\in\Omega} \phi\Big(\|\nabla \boldsymbol{U}_x(\boldsymbol{t}_i)\|_2^2 + \|\nabla \boldsymbol{U}_y(\boldsymbol{t}_i)\|_2^2\Big) + \beta \sum_{i\in\Omega} \phi\Big(\|\nabla \boldsymbol{V}_x(\boldsymbol{t}_i)\|_2^2 + \|\nabla \boldsymbol{V}_y(\boldsymbol{t}_i)\|_2^2\Big), \tag{4}$$

where $\beta = 0.1$ is the weight coefficient for this cost, $\nabla$ denotes the gradient operator, and $\boldsymbol{U}_x$, $\boldsymbol{U}_y$ denote the horizontal and vertical components of the motion field $\boldsymbol{U}$, respectively (similarly for $\boldsymbol{V}_x$ and $\boldsymbol{V}_y$). This cost penalizes large magnitudes in the gradient of the flow field with a robust function $\phi(x)$ and regularizes the flow vectors between the synthesized colors in adjacent frames.

### 4.3 Optimization

Since Eq. 1 is non-convex, we use an iterative optimization algorithm that alternates between minimizing different subsets of the variables. We apply the optimization in a coarse-to-fine manner to reduce the chance of prematurely locking into a bad local minimum. At each iteration, we first fix the colors $\boldsymbol{I}$ and flow fields $\boldsymbol{U},\boldsymbol{V}$, and solve for source patch position and transformation $(\boldsymbol{s}_i,\theta_i)$ for each target patch $\boldsymbol{t}_i$. This corresponds to the patch search step for estimating the dense approximate nearest neighbor field $\{\boldsymbol{s}_i,\theta_i \mid i\in\Omega\}$. We then fix the estimated nearest neighbor field $(\boldsymbol{s}_i,\theta_i)$ and flow fields $\boldsymbol{U},\boldsymbol{V}$ and estimate colors $\boldsymbol{I}$. This color synthesis step minimizes local appearance differences between the source and target patches spatially and the color differences along flow vectors temporally. Finally, we fix the synthesized colors $\boldsymbol{I}$ and update the forward/backward flow fields $\boldsymbol{U},\boldsymbol{V}$. Algorithm 1 summarizes this procedure in pseudocode. We now describe each step in detail.

**Nearest neighbor field estimation** Given the currently estimated color $\boldsymbol{I}$ and forward/backward flow fields $\boldsymbol{U},\boldsymbol{V}$, we want to search for each target pixel position $\boldsymbol{t}_i$ (1) the source position $\boldsymbol{s}_i$, and (2) the geometric patch transformation $\theta_i$ that minimizes the overall objective in Eq. 1. This is equivalent to just minimizing the first term of the objective function Eq. 2. To this end we extend the generalized PatchMatch algorithm [Barnes et al. 2010] to the spatio-temporal case. We initialize the source patch position $\boldsymbol{s}_i$ and patch transformation $\theta_i$ through random sampling, and then update the estimation by alternating between the random search and propagation steps.

Random search: At this step, we generate a sequence of random samples $(\boldsymbol{s}_i,\theta_i)$ (i.e., source position, rotation and scale) from an exponential distribution [Barnes et al. 2010]. We update the nearest neighbor field if any of the randomly selected sample achieves a lower cost.

Spatial and temporal propagation: This step involves propagating good nearest neighbor candidates spatially and temporally. For the spatial propagation, we follow the generalized PatchMatch algorithm to take into account the geometric transformation of the source patch. The temporal propagation step, however, has two important differences regarding candidate patch position and transformation:

*(1) Propagating patch position:* In a straightforward extension of the PatchMatch algorithm (e.g., as in [Newson et al. 2014]), for the target patch centered at $(t_i^x, t_i^y, t_i^f \pm 1)$, the algorithm would consider candidate source patches centered at $(s_i^x, s_i^y, s_i^f \pm 1)$. However, this temporal propagation strategy implicitly assumes that the motion at the target pixel $\boldsymbol{t}_i$ is the same as the motion at the source pixel $\boldsymbol{s}_i$. We avoid this by temporally propagating the candidate patches along the estimated forward and backward flow vectors; see Figure 4 as illustration of forward temporal propagation. For target patch $\boldsymbol{t}_i$, we generate the candidate source patch position $\boldsymbol{s}_i$ by: (1) finding temporal neighbor of $\boldsymbol{t}_i$ using backward flow: $\boldsymbol{t}_i' = \boldsymbol{t}_i + \boldsymbol{V}(\boldsymbol{t}_i)$, (2) finding the corresponding source patch $\boldsymbol{s}_i'$ of the target patch $\boldsymbol{t}_i'$ from the current nearest neighbor field, and (3) propagating the source patch to the next frame using forward flow $\boldsymbol{s}_i = \boldsymbol{s}_i' + \boldsymbol{U}(\boldsymbol{s}_i)$. The backward temporal propagation follows a similar procedure. The flow-guided temporal propagation removes the assumption that the motion must be consistent in the source and target patches.

*(2) Propagating patch transformation:* We observe that the local patches of flow vectors provide cues of the *patch transformation* from the current frame to the adjacent frames. For example, the apparent size of an object increases under camera zoom. Scaling of patches can be inferred from the relative densities of flow vectors from frame to frame. More specifically, we warp a grid of pixel positions using optic flow and estimate the propagated candidate transformation matrix as $T(\boldsymbol{U}(\boldsymbol{s}_i'))\,S(\theta_i')\,T(\boldsymbol{V}(\boldsymbol{t}_i))$, where $S(\theta_i')$ is the transformation matrix of source patch $\boldsymbol{s}_i'$ and $T(\boldsymbol{V}(\boldsymbol{t}_i))$ is the estimated patch transformation matrix using local flow vectors at position $\boldsymbol{t}_i$. $T(\boldsymbol{U}(\boldsymbol{s}_i'))$ is similarly defined.

[2] Note that we define the Charbonnier penalty as $\phi(x^2)$ with the intention of providing more clarity and consistency in our formulation.

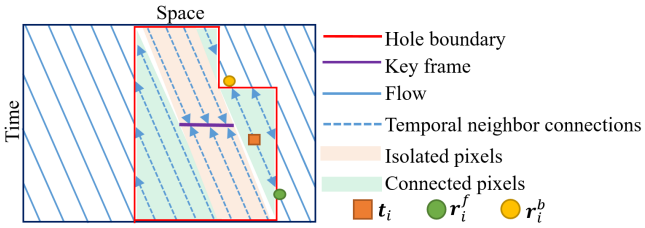



**Figure 5:** *Flow-guided color synthesis. For all target pixels, we find their temporal neighbors in the source regions by traversing the estimated flow vectors. We then use these temporal neighbors to enforce temporal coherence.*

**Flow-guided color synthesis** Given the currently estimated forward/backward flow fields $\boldsymbol{U},\boldsymbol{V}$, we estimate the color in the missing regions by minimizing $\boldsymbol{E}_{color\text{-}spatial}+\boldsymbol{E}_{color\text{-}temporal}$.

The spatial color cost $\boldsymbol{E}_{color\text{-}spatial}$ by itself could be minimized using standard patch voting, i.e., computing the weighted average of the overlapping source patches. Incorporating the temporal color cost $\boldsymbol{E}_{color\text{-}temporal}$ turns the problem into a 3D Poisson equation with temporal connections specified by the forward/backward flow vectors. It can be solved with the Gauss-Seidel method. Let $\boldsymbol{Z}$ denote the color field obtained by spatial voting, i.e., the minimizer of $\boldsymbol{E}_{color\text{-}spatial}$. Applying the Gauss-Seidel method involves iteratively averaging $\boldsymbol{Z}$ and the warped images from the forward/backward flow. However, we found that this algorithm requires many iterations to convergence, particularly when the hole contains a long temporal span. In addition, the repeated application of bicubic interpolation (for sampling colors in sub-pixel positions) introduces unwanted blur and ringing artifacts.

We propose a heuristic to address this issue. First, we categorize the missing pixels into two classes:

**Connected pixels** are pixels that are visible somewhere in the video. By following their forward or backward flow connections *transitively* we eventually reach a known pixel. We call these known pixels the transitive temporal neighbors of the connected pixel, designated $\boldsymbol{r}_i^f$ (forward) and $\boldsymbol{r}_i^b$ (backward) for the $i^{th}$ pixel. For connected pixels we directly penalize deviation from the transitive temporal neighbor colors, avoiding error accumulation from repeated resampling.

**Isolated pixels** are not transitively connected to any known pixels. It is difficult to synthesize isolated pixels consistently using the Gauss-Seidel solver described above. To address this issue we find the frame that has most isolated pixels and designate it a *key frame*. The isolated pixels in the key frame are treated as if they are known pixels, i.e., they become the transitive temporal neighbors for other isolated pixels, turning them into connected pixels. We greedily pick key frames in this manner until all isolated pixels are connected. The key frame pixels themselves do not have temporal neighbors, and use spatial voting only. In a way they are synthesized like in a standard *single-image* completion problem, albeit jointly solved in a video completion problem.

As both connected and isolated pixels are related to the unknown backward/forward flow fields, we re-assign connected and isolated pixels after we update the forward/background flow fields in each iteration.

Figure 5 illustrates the situations described above. Formally, the heuristic above is implemented by minimizing the following objective:

$$\underset{\boldsymbol{I},\boldsymbol{Z}}{\operatorname{argmin}} \sum_{i\in\overline{\Omega}} \|\boldsymbol{I}(\boldsymbol{t}_i)-\boldsymbol{Z}(\boldsymbol{t}_i)\|_2^2+\alpha\phi\left(\left\|\boldsymbol{I}(\boldsymbol{t}_i)-\boldsymbol{I}(\boldsymbol{r}_i^f)\right\|_2^2\right) \quad (5)$$
$$+\alpha\phi\left(\left\|\boldsymbol{I}(\boldsymbol{t}_i)-\boldsymbol{I}(\boldsymbol{r}_i^b)\right\|_2^2\right).$$

For pixels that are missing either $\boldsymbol{r}_i^f$ or $\boldsymbol{r}_i^b$ we drop the respective terms (isolated pixels in key frames do not have temporal neighbors, all other pixels have at least one temporal neighbor).

As the objective function in Eq. 5 decomposes into $|\overline{\Omega}|$ independent loss functions, we can drop the pixel index $i$ for simplicity. However, these loss functions are non-linear and the problem cannot be solved directly. We implement a gradient-descent scheme that iteratively converges to a local minimizer. We use the spatial voting results $\boldsymbol{Z}$ as an initialization, which corresponds to only minimizing spatial cost and ignoring temporal coherence. We then iteratively solve the best increment $\mathrm{d}\boldsymbol{I}$ by setting the derivative of the objective function to zero. Denote the differences of the $\boldsymbol{Z}(\boldsymbol{t}_i)-\boldsymbol{I}(\boldsymbol{t}_i)$ as $\mathrm{d}\boldsymbol{Z}$, $\mathbf{I}(\boldsymbol{r}_i^f)-\boldsymbol{I}(\boldsymbol{t}_i)$ as $\mathrm{d}\boldsymbol{I}^f$, and $\mathbf{I}(\boldsymbol{r}_i^b)-\boldsymbol{I}(\boldsymbol{t}_i)$ as $\mathrm{d}\boldsymbol{I}^b$. In each iteration, we can compute the optimal increment in close-form:

$$\mathrm{d}\boldsymbol{I}(\boldsymbol{t}_i)=\frac{1}{C}\left[\mathrm{d}\boldsymbol{Z}+\alpha\left(\phi'\left((\mathrm{d}\boldsymbol{I}^f)^2\right)\mathrm{d}\boldsymbol{I}^f+\phi'\left((\mathrm{d}\boldsymbol{I}^b)^2\right)\mathrm{d}\boldsymbol{I}^b\right)\right], \quad (6)$$

$$C=1+\alpha\left(\phi'\left((\mathrm{d}\boldsymbol{I}^f)^2\right)+\phi'\left((\mathrm{d}\boldsymbol{I}^b)^2\right)\right), \quad (7)$$

where $\phi'(x^2)=\frac{1}{\sqrt{x^2+\varepsilon^2}}$ is the first-order derivative of the robust function $\phi(\cdot)$. We add the optimal increment $\mathrm{d}\boldsymbol{I}(\boldsymbol{t}_i)$ back to $\boldsymbol{I}(\boldsymbol{t}_i)$ and solve Eq. 7 in the next iteration. We find that five iterations are sufficient for convergence. Solving the modified objective function in Eq. 5 strikes a good balance between temporal consistency and spatial coherence.

**Forward and backward flow field estimation** In this step, we fix the color $\boldsymbol{I}$ and refine and update the forward/backward flow fields $\boldsymbol{U},\boldsymbol{V}$. The minimization problem corresponds to a typical two-frame optical flow estimation problem with a data term and a spatial regularization term. We use the previously estimated flow as initialization and iteratively estimate the flow increments. We use the Iterative Reweighted Least Squares (IRLS) formulation [Liu 2009] to compute the forward and background flow field in the target regions (our method is not tied to this particular optical flow algorithm, any other state-of-the-art algorithm could be used instead).

### 4.4 Initialization

To bootstrap the optimization process we need to initialize color, flow, and nearest neighbor fields at the coarsest scale. We first initialize the nearest neighbor field $\{\boldsymbol{s}_i,\theta_i\}$ with random samples, and the flow fields $\boldsymbol{U},\boldsymbol{V}$ by smoothly interpolating the known values at the boundary inward (independently for each frame). Next, we initialize the colors $\boldsymbol{I}$ by running the nearest neighbor field estimation and the flow-guided color synthesis step, and finally update the flow fields using the synthesized colors.

### 4.5 Implementation details

We use relatively small $5\times5$ patches in our algorithm. While typical image completion algorithms use larger patches, e.g., $10\times10$ [Darabi et al. 2012], we observe that increasing the patch size does not substantially improve the visual quality of the results while significantly increasing the runtime. We attribute this effect to the additional temporal connections by the forward and backward flow

vectors. These connections help efficiently propagate good candidates along the temporal dimension for nearest neighbor search.

To enforce that all pixels in $Q_i$ fall in $\Omega$, we compute the 2D distance transform of the hole mask for each frame, and reject any source patch $Q_i$ whose distance (as measured from the patch center $s_i$) is smaller than the patch radius. For example, for a $5\times5$ patch with no scaling, the patch radius is $\frac{5\sqrt{2}}{2}$.

We set the color temporal cost weight $\alpha = 0.5$ and the flow spatial weight $\beta = 0.1$. We fix the weights for all our experiments.

We optimize the objective function in a multi-resolution fashion. We set the number of image pyramid levels such that the height at the coarse scale is between 32 and 64 pixels. At the coarsest scale, we run 20 iterations and decrease iteration count in each subsequent scale by 4 (though we never use less than 4 iterations).

Our algorithm relies on the estimated flow to enforce temporal consistency. Optical flow algorithms are far from perfect, however, particularly in occluded/dis-occluded regions. We are thus interested in identifying unreliable flow pixels. While existing learning-based flow confidence measures [Mac Aodha et al. 2013] appear to be robust, they are computationally intensive. Instead, we use forward-backward flow consistency to identify unreliable flow. Specifically, we compute confidence scores $c_i^f, c_i^b$ for forward and backward flow,

$$c_i^f = \exp\left(-\frac{\|\boldsymbol{U}(\boldsymbol{t}_i)+\boldsymbol{V}(\boldsymbol{t}_i+\boldsymbol{U}(\boldsymbol{t}_i))\|_2^2}{2\sigma_F^2}\right), \quad (8)$$

$$c_i^b = \exp\left(-\frac{\|\boldsymbol{V}(\boldsymbol{t}_i)+\boldsymbol{U}(\boldsymbol{t}_i+\boldsymbol{V}(\boldsymbol{t}_i))\|_2^2}{2\sigma_F^2}\right), \quad (9)$$

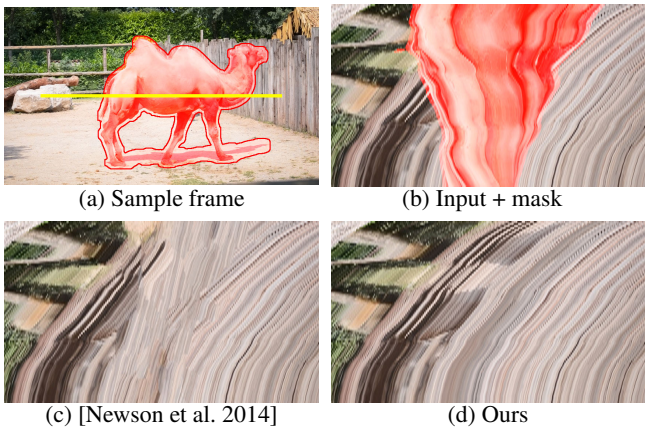
where $\sigma_F = 1$ controls the sensitivity. We label the forward flow $\boldsymbol{U}(\boldsymbol{t}_i)$ at $\boldsymbol{t}_i$ as "unreliable" when the confidence score $c_i^f < 0.5$, and similar for the backward flow. We discard these unreliable flow vectors when searching for the temporal neighbors of the missing pixels. As a result, we drop the respective terms in Eq. 5 when performing the flow-guided color update, preventing the influence by unreliable flows.

## 5 Results

We implement the completion algorithm in MATLAB. For optical flow computation, we use the C++ implementation from Liu [2009]. Processing a short video with $854 \times 480$ pixels and 90 frames and with a moderately sized missing region (e.g., CAMEL: 6.5M = 17.81% of pixels missing) our non-optimized implementation took around 3 hours on a desktop computer with 2.8 GHz Intel i7 CPU (quad-core) and 12GB memory.

**Removing dynamically moving objects in natural scenes** We evaluated our algorithm on a variety of challenging sequences from a recent benchmark dataset [Perazzi et al. 2016]. While the dataset was intended for evaluating video object segmentation algorithms, the image sequences present multiple instances of challenges for evaluating video completion algorithms "in the wild." The major challenges including dynamic background, motion blur, camera shake, background clutter, and complicated hole shapes. We dilate the ground truth pixelwise annotation using a $15 \times 15$ structuring element. We then use the RotoBrush tool in Adobe AfterEffect to include cast shadows in the mask.

Figure 6 shows sample frames from five input image sequences with mask overlay (odd rows) and our completion results (even rows). In the first three sequences CAMEL, BREAKDANCE, and KITE-SURF, we demonstrate that our algorithm can seamlessly fill the missing dynamic background for videos captured with freely moving camera. The HORSEJUMP-LOW and FLAMINGO sequences highlight the advantage of our flow-guided color synthesis for accurately propagating known colors into the hole.



(a) Sample frame (b) Input + mask

(c) [Newson et al. 2014] (d) Ours

**Figure 9:** *Temporally coherent completion. We take the sequence* CAMEL *and visualize the completion results using spatiotemporal x-t slice of the video along the profile (yellow line) in (a). (b) The x-t slice of the video with mask marked as red. (c) Results from [Newson et al. 2014]. (d) Our results. We can clearly see that the completion results in (c), while seeming locally plausible, fail to maintain long-term temporal consistency. The combination of patch-based optimization and dense flow field allows us to preserve the temporal continuity with high spatial frequency.*

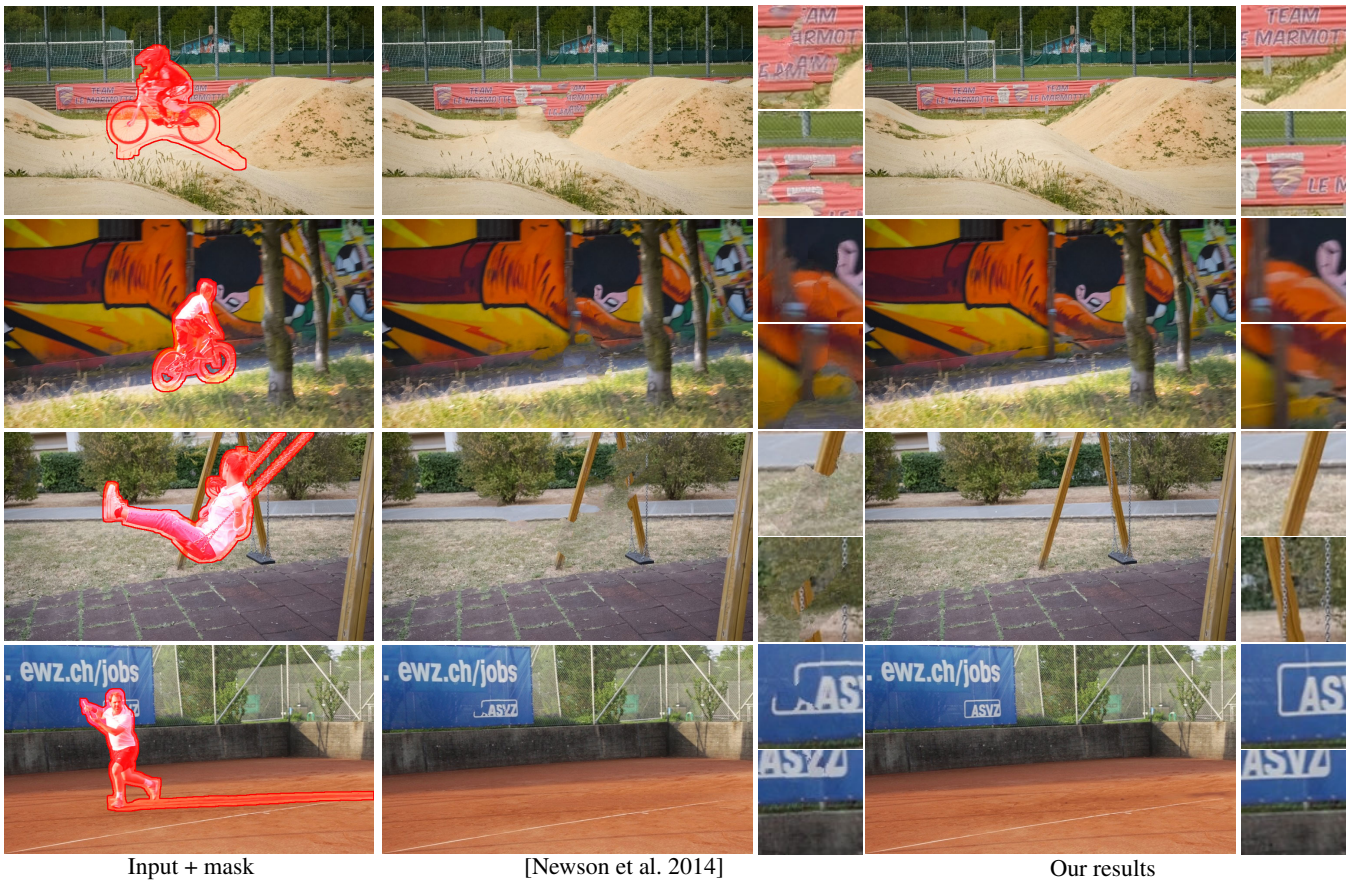
**Comparisons with state-of-the-art methods** We qualitatively compare our method with a recent state-of-the-art patch-based video completion algorithm [Newson et al. 2014]. We used the code released by the author and tested it on the image sequences [Perazzi et al. 2016] using the default parameters provided by the authors. Figure 7 shows the representative frames from four sequences and the completion results. Newson et al. [2014] fill the hole by sampling spatio-temporal patches from source regions. We can see that such a technique introduces severe artifacts because of the inconsistent motion between the source and target regions. Our method, on the other hand, fills the missing regions with convincing content.

In Figure 8, we compare with a segmentation-based techniques for background [Granados et al. 2012a] and foreground inpainting [Granados et al. 2012b]. As the code is not publicly available, we compare with them using sequences from their papers. Our method achieves comparable quality without the need to provide dense pixel-wise mask of the foreground objects [Granados et al. 2012a] or spatio-temporal search regions [Granados et al. 2012b].

In Figure 9, we highlight the temporal coherence aspect of the completion results. In Figure 9(b), we show a spatio-temporal x-t slice of the video along the marked profile. In Figure 9(c–d), we show the x-t slice of the *completed* video by [Newson et al. 2014] and our approach, respectively. From the x-t slice visualization we can clearly see that our results are temporally coherent and are adapted to the non-trivial camera motion in the known regions.

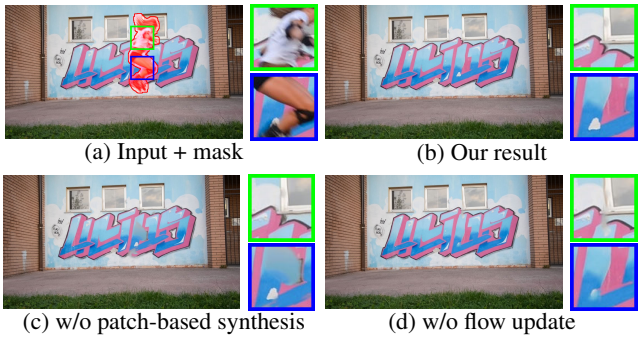
**Contributions of each component** We evaluate the contributions of each of the major components to the final performance in Figure 10. We tested the sequence ROLLERBLADE by disabling (1) patch-based optimization (i.e., propagate colors from the known

**Figure 6:** *Object removal from video sequences* CAMEL, BREAKDANCE, KITE-SURF, HORSEJUMP-LOW, *and* FLAMINGO. *For each input sequence (odd row), we show representative frames with mask overlay. We show the completed results in even rows.*




Input + mask [Newson et al. 2014] Our results

**Figure 7:** *Comparison to [Newson et al. 2014] on sequences* BMX-BUMPS, BMX-TREES, SWING, *and* TENNIS. *These sequences are challenging due to the motion blur from the fast camera motion. Our algorithm seamlessly removes the dynamic object under shaky motion. Newson et al. [2014], on the other hand, produces visible artifacts spatially and fails to generate temporally coherent results.*



(a) Input + mask (b) Our result

(c) w/o patch-based synthesis (d) w/o flow update

**Figure 10:** *Contribution of different components of the proposed algorithm to the final results. (b) Our result. (c) Without patch-based synthesis, the algorithm cannot hallucinate regions that are not visible in the image sequence (see the blue box). (d) Disabling the flow update introduces visible artifacts.*

boundary into the hole) and (2) flow estimation (i.e., using interpolated flow only), respectively.

**Comparison to single image completion algorithms** Figure 11 shows the advantage of video completion (or multi-frame image completion) over conventional image completion from a single image. In the sequence DANCE-JUMP, searching for usable texture from the same source image fails to complete the missing region with plausible contents. We show in Figure 11 results from two state-of-the-art image completion algorithms: Photoshop Content Aware Fill and [Huang et al. 2014]. Our video completion result (top-right) shows a convincing completion by transferring available contents that are visible in other frames in the sequence.

**Limitations** Our approach has several limitations.

Computational complexity: The computational complexity of the proposed algorithm is high and the runtime is far from being at interactive rates. Our current implementation is impractical for many video editing applications. In our experiments, about 65% of the overall computation is spent on iteratively computing and refining the forward and backward flow fields in the image sequence. Using GPUs to speed up the flow computation may help speed up the completion process. Note that our approach is not tied to any particular flow algorithm, so we could easily switch to a more advanced optical flow algorithm. Another promising direction is to adopt PatchTable [Barnes et al. 2015] to significantly reduce the run-time of the nearest neighbor patch search step. However, several modifications may be required, including generalization to affine transformed patches and incorporation of forward/backward flow connections into the k-coherence step.

Noticeable artifacts: While we considerably expand the range of input videos for completion techniques applicable to videos where the camera, foreground, and background all are dynamic, artifacts

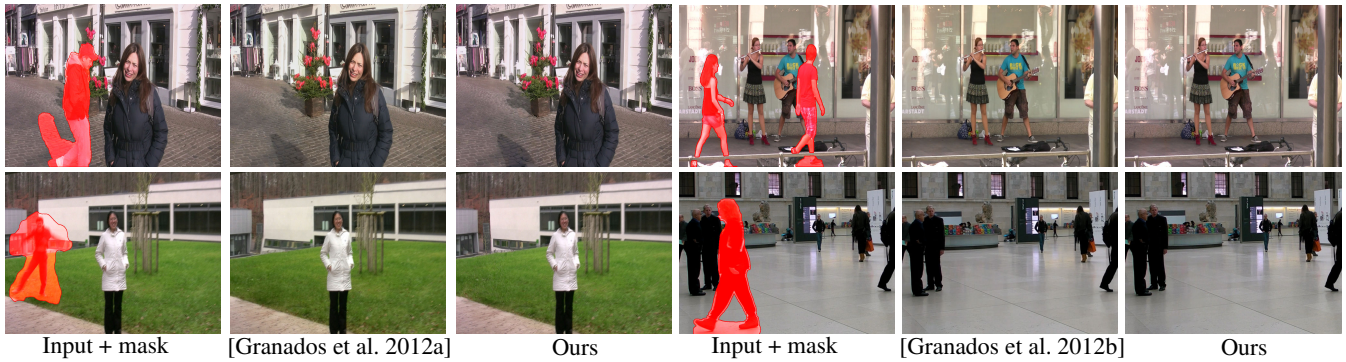



**Figure 8:** *Comparison to segmentation-based methods for background [Granados et al. 2012a] and foreground [Granados et al. 2012b] inpainting on sequences from their papers. Our approach achieves similar visual quality without the need of manually segmenting out the dynamic foreground objects in the scene or manually specifying search regions.*

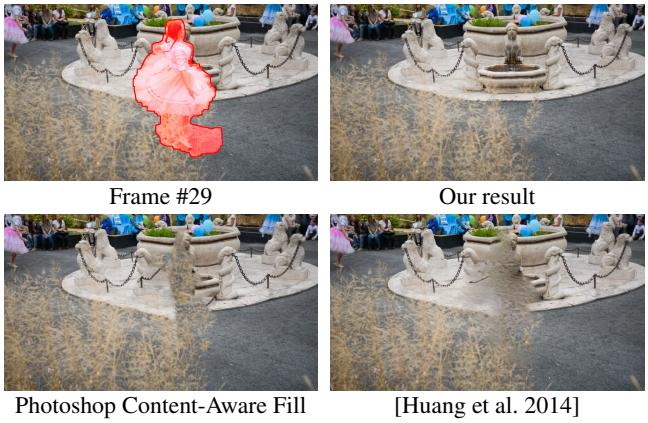



**Figure 11:** *The advantage of video completion over image completion algorithms. Our video completion algorithm faithfully recover the missing region by taking all the frames into consideration.*

**Figure 12:** *Our algorithm may fail to hallucinate large missing areas. Here the artifacts are visible with a closer examination.*

remain visible in some of our results. We attribute these artifacts mainly to the following reasons:

1. The dense flow field estimation is sometimes unreliable for dynamic texture such as water surfaces. Since our algorithm relies on accurate flow to guide the completion it can produce visible artifacts in these cases (e.g., Figure 13 and the BOAT scene).

2. For highly structured scenes with large depth variation our algorithm sometimes fails to synthesize flow fields that respect the scene geometry (e.g., PARKOUR scene).

3. When a large area is occluded throughout the entire sequence (i.e., a large number of isolated pixels), the performance of our algorithm is limited by how well a single-image completion algorithm can fill the hole. While our completion will be temporally coherent, spatial artifacts may be visible at closer examination (e.g., ELEPHANT scene, Figure 12).


**Figure 13:** *Comparison with [Newson et al. 2014] on video sequences from their paper. Our approach produces reason results but has more visible artifacts than [Newson et al. 2014] due to incorrect flow synthesis.*

Handling complex foreground motion: In the case of a *static* camera, such as the BREAKDANCE video, Newson et al. [2014] may synthesize complex motion (e.g., clapping hands) more successfully than our method by exploiting non-local repetition of spatio-temporal patches. The advantage of our method that uses spatial-only patches is that it is applicable to videos with a *dynamically moving* (e.g., hand-held) camera. Exploiting multi-resolution patches in the temporal domain may combine the best of two worlds and help improve the temporal consistency in both moving and static camera scenarios. This will be an interesting direction for future work.

Mask selection: In this paper, we use user-specified foreground mask as our input. In practice, interactive or automatic video object segmentation remains a challenging problem and an active research field. In the STROLLER sequence, we show a failure case where the soft shadows and reflections of the moving object are not included in the user-specified mask, therefore our algorithm cannot remove them. It is up to the user to include all affected pixels in the mask if it is desired to remove all traces of an object.

## 6 Conclusions

In this paper, we presented a robust video completion algorithm that is capable of handling a wide variety of challenging scenarios. Our main contribution is to combine the advantages of patch-based optimization framework and pixel-wise flow field representation for

temporally coherent synthesis. We formulate the filling process as a global optimization of color and flow and present an alternating optimization approach to minimize the objective function. Experimental results show that our approach significantly extends the capability of video completion to videos containing multiple dynamic objects, scene depth variations, and captured with a moving camera.

## Acknowledgements

J.-B. Huang and N. Ahuja are supported in part by Office of Naval Research under Grant N00014-16-1-2314.

## References

BARNES, C., SHECHTMAN, E., FINKELSTEIN, A., AND GOLDMAN, D. 2009. PatchMatch: a randomized correspondence algorithm for structural image editing. *ACM Trans. on Graphics 28*, 3, 24.

BARNES, C., SHECHTMAN, E., GOLDMAN, D., AND FINKELSTEIN, A. 2010. The generalized patchmatch correspondence algorithm. In *ECCV*.

BARNES, C., ZHANG, F.-L., LOU, L., WU, X., AND HU, S.-M. 2015. PatchTable: Efficient patch queries for large datasets and applications. *ACM Trans. on Graphics 34*, 4, 97.

BHAT, K. S., SEITZ, S. M., HODGINS, J. K., AND KHOSLA, P. K. 2004. Flow-based video synthesis and editing. In *ACM Trans. on Graphics*, vol. 23, ACM, 360–363.

CRIMINISI, A., PÉREZ, P., AND TOYAMA, K. 2004. Region filling and object removal by exemplar-based image inpainting. *IEEE TIP 13*, 9, 1200–1212.

DARABI, S., SHECHTMAN, E., BARNES, C., GOLDMAN, D. B., AND SEN, P. 2012. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Trans. on Graphics 31*, 4, 82.

DRORI, I., COHEN-OR, D., AND YESHURUN, H. 2003. Fragment-based image completion. In *ACM Trans. on Graphics*, vol. 22, 303–312.

EFROS, A. A., AND LEUNG, T. K. 1999. Texture synthesis by non-parametric sampling. In *ICCV*, vol. 2.

GRANADOS, M., KIM, K. I., TOMPKIN, J., KAUTZ, J., AND THEOBALT, C. 2012. Background inpainting for videos with dynamic objects and a free-moving camera. In *ECCV*.

GRANADOS, M., TOMPKIN, J., KIM, K., GRAU, O., KAUTZ, J., AND THEOBALT, C. 2012. How not to be seenobject removal from videos of crowded scenes. In *Computer Graphics Forum*, vol. 31, 219–228.

GUILLEMOT, C., AND LE MEUR, O. 2014. Image inpainting: Overview and recent advances. *IEEE Signal Processing Magazine 31*, 1, 127–144.

HUANG, J.-B., KANG, S. B., AHUJA, N., AND KOPF, J. 2014. Image completion using planar structure guidance. *ACM Trans. on Graphics 33*, 4, 129.

ILAN, S., AND SHAMIR, A. 2015. A survey on data-driven video completion. *Computer Graphics Forum 34*, 6, 60–85.

JAMRIŠKA, O., FIŠER, J., ASENTE, P., LU, J., SHECHTMAN, E., AND SỲKORA, D. 2015. LazyFluids: Appearance transfer for fluid animations. *ACM Trans. on Graphics 34*, 4, 92.

JIA, J., TAI, Y.-W., WU, T.-P., AND TANG, C.-K. 2006. Video repairing under variable illumination using cyclic motions. *IEEE TPAMI 28*, 5, 832–839.

KALANTARI, N. K., SHECHTMAN, E., BARNES, C., DARABI, S., GOLDMAN, D. B., AND SEN, P. 2013. Patch-based high dynamic range video. *ACM Trans. on Graphics 32*, 6, 202–1.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. In *ACM Trans. on Graphics*, vol. 22, 277–286.

KWATRA, V., ESSA, I., BOBICK, A., AND KWATRA, N. 2005. Texture optimization for example-based synthesis. In *ACM Trans. on Graphics*, vol. 24, 795–802.

LIU, C., AND FREEMAN, W. T. 2010. A high-quality video denoising algorithm based on reliable motion estimation. In *ECCV*.

LIU, C. 2009. *Beyond pixels: Exploring new representations and applications for motion analysis*. PhD thesis.

MAC AODHA, O., HUMAYUN, A., POLLEFEYS, M., AND BROSTOW, G. J. 2013. Learning a confidence measure for optical flow. *IEEE TPAMI 35*, 5, 1107–1120.

MATSUSHITA, Y., OFEK, E., GE, W., TANG, X., AND SHUM, H.-Y. 2006. Full-frame video stabilization with motion inpainting. *IEEE TPAMI 28*, 7, 1150–1163.

NEWSON, A., ALMANSA, A., FRADET, M., GOUSSEAU, Y., PÉREZ, P., ET AL. 2014. Video inpainting of complex scenes. *SIAM Journal on Imaging Sciences*.

PATWARDHAN, K. A., SAPIRO, G., AND BERTALMIO, M. 2005. Video inpainting of occluding and occluded objects. In *ICIP*.

PATWARDHAN, K. A., SAPIRO, G., AND BERTALMÍO, M. 2007. Video inpainting under constrained camera motion. *IEEE TIP 16*, 2, 545–553.

PERAZZI, F., PONT-TUSET, J., MCWILLIAMS, B., GOOL, L. V., GROSS, M., AND SORKINE-HORNUNG, A. 2016. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*.

PRITCH, Y., KAV-VENAKI, E., AND PELEG, S. 2009. Shift-map image editing. In *ICCV*.

ROXAS, M., SHIRATORI, T., AND IKEUCHI, K. 2014. Video completion via spatio-temporally consistent motion inpainting. *Information and Media Technologies 9*, 4, 500–504.

SEVILLA-LARA, L., WULFF, J., SUNKAVALLI, K., AND SHECHTMAN, E. 2015. Smooth loops from unconstrained video. *Computer Graphics Forum 34*, 4, 99–107.

SHECHTMAN, E., RAV-ACHA, A., IRANI, M., AND SEITZ, S. 2010. Regenerative morphing. In *CVPR*.

SHIRATORI, T., MATSUSHITA, Y., TANG, X., AND KANG, S. B. 2006. Video completion by motion field transfer. In *CVPR*.

STROBEL, M., DIEBOLD, J., AND CREMERS, D. 2014. Flow and color inpainting for video completion. In *German Conference on Pattern Recognition*.

WEXLER, Y., SHECHTMAN, E., AND IRANI, M. 2007. Space-time completion of video. *IEEE TPAMI 29*, 3, 463–476.

XUE, T., RUBINSTEIN, M., LIU, C., AND FREEMAN, W. T. 2015. A computational approach for obstruction-free photography. *ACM Trans. on Graphics 34*, 4, 79.