Consistent Video Filtering for Camera Arrays

Nicolas Bonneel, James Tompkin, Deqing Sun, Oliver Wang, Kalyan Sunkavalli, Sylvain Paris, Hanspeter Pfister

EUROGRAPHICS 2017 / L. Barthe and B. Benes
(Guest Editors)

Volume 36 (2017), Number 2

Consistent Video Filtering for Camera Arrays

Nicolas Bonneel[*,1], James Tompkin[*,2], Deqing Sun[3], Oliver Wang[4], Kalyan Sunkavalli[4], Sylvain Paris[4], and Hanspeter Pfister[5]

[*] Equal contribution, [1]CNRS/LIRIS, [2]Brown University, [3]NVIDIA, [4]Adobe, [5]Harvard Paulson SEAS

(a) *Input camera array video (stereo rig).* **(b)** *Per-frame filtered with dehaze operator.* **(c)** *Our result after enforcing consistency.*

Figure 1: *Our method turns inconsistent filtered video from camera arrays into* consistent *video across time and view. Applying image dehazing per frame to an input stereo video (a) produces unstable results across time and view (b). Our approach makes the resulting stereo video consistent (c). ('Up and Down':* $\lambda_t = 0.05, \lambda_s = 0.05$.*)*

Abstract

Visual formats have advanced beyond single-view images and videos: 3D movies are commonplace, researchers have developed multi-view navigation systems, and VR is helping to push light field cameras to mass market. However, editing tools for these media are still nascent, and even simple filtering operations like color correction or stylization are problematic: naively applying image filters per frame or per view rarely produces satisfying results due to time and space inconsistencies. Our method preserves and stabilizes filter effects while being agnostic to the inner working of the filter. It captures filter effects in the gradient domain, then uses input *frame gradients as a reference to impose temporal and spatial consistency. Our least-squares formulation adds minimal overhead compared to naive data processing. Further, when filter cost is high, we introduce a filter transfer strategy that reduces the number of per-frame filtering computations by an order of magnitude, with only a small reduction in visual quality. We demonstrate our algorithm on several camera array formats including stereo videos, light fields, and wide baselines.*

Categories and Subject Descriptors (according to ACM CCS): I.4.3 [Computer Graphics]: Enhancement—Filtering I.2.10 [Computer Graphics]: Vision and Scene Understanding—Video analysis

1. Introduction

Capturing video with multiple cameras is an integral component of many popular applications. For example, camera array rigs record content for 3D displays, for stitched 360° video for virtual reality [PSZ*15], for virtual-camera-based post production [FK00], and for refocusing and depth-aware processing [NLB*05, VLD*13].

However, one challenge lies in processing the captured footage, as many methods produce inconsistent output across frames from multiple views. Consider the large class of filtering operations—from tone mapping to color grading—which are fundamental steps in post production workflows. With naive processing, small content changes between frames can cause jarring temporal and spatial inconsistency artifacts. We address filter inconsistency and enforce spatio-temporal consistency across multiple views, with a gradient-domain optimization inspired by Bonneel et al. [BTS*15].

A second challenge is efficient filtering. Camera arrays often produce 10–100× more video data than single-view cameras, and so *any* kind of filtering is expensive, both in terms of computation and memory. As an example, a 91-view sequence at 1280 × 768 resolution contains about 8GB of data *per second*. We present an efficient approach that reduces the cost of filtering the data by an order of magnitude, without a large reduction in quality, by skipping the filtering step on many frames completely, and instead, *transferring* the filter response from a small subset of input frames to the rest of

the data. Further, our approach is causal as it sequentially processes frames, and so does not need multiple passes, or even to keep multiple views or entire videos in memory. Combined, these properties reduce the computational burden and allow for of spatio-temporally consistent filtering of camera array video.

1.1. Related Work

Some techniques allow users to make consistent local edits across a few discrete views [SK02, WJYG08, YJHS12, LBP*12], in a video [RAKRF08, BPK*13, SSML14], or in dense light fields [ZWS*16, JMG11, JMB*14, MJG14, AZJ*15]. The challenge tackled by these techniques is different from ours as the effect is essentially a painting or pasting operation: once the scene location is determined, the edit is inherently stable across views. Similarly, due to the popularity of stereo video footage, many works deal with explicitly enforcing consistency across stereo views for operations like image stabilization [LNJ13], disparity mapping [LHW*10, LSS*15], hole filling [MHCP12] or object copy-and-paste [LvBK*10, LSS*15].

In contrast, we consider the class of image filter edits. Many filters are sensitive to small changes in an image, and may generate inconsistent results over time and view when applied to video from camera arrays. For instance, recent intrinsic decomposition methods for static light field images address the need for consistency across views [GEZ*16, AG16]. Unlike methods that target a specific effect for a given input format, we propose an approach that applies to a large number of image filters across different camera array setups.

One similar application is to harmonize colors across images in a collection [HSGL13]. These photographic edits can typically be modeled by a global RGB remapping. While we share a reliance on dense matching between images, we seek to address a larger class of filters including sophisticated spatially-varying transforms such as intrinsic decomposition. This requires us to develop a number of modifications to a dense matching method—in our case, PatchMatch [BSFG09]—that enable us to handle wider view separation while retaining computational efficiency. In time, our approach is also causal, which helps it scale to light field videos.

Other techniques target flickering in videos, i.e., due to physical degradation of film stock [BZCC10, DD10, PKCK06] or poor automatic white balance [FL11]. These techniques seek to remove undesirable artifacts from degraded input video. In comparison, our input medium is *clean*, and we wish to apply an image filter. We seek to remove the flickering generated by the filter while still preserving its visual effect. For instance, for tone mapping, we remove the inconsistency between frames but maintain the low dynamic range. Some processing techniques have similar goals and are effective at fixing temporal instabilities [LWA*12, BTS*15], even in pixel prediction maps [DYY15]. However, they focus on single-view sequences and do not cope with view inconsistencies, e.g., when processing light fields. Our approach revisits that of Bonneel et al. to ensure spatial and temporal consistency for different camera array setups. Importantly, this allows us to introduce the notion of filter transfer to speed up the per-frame filtering.

1.2. Contributions

For camera array datasets, which includes stereo and light field images and videos, we contribute:

- A method to remove spatial and temporal inconsistencies caused by applying unstable image filters.
- An efficient least-squares formulation of this method that can be solved in linear time with respect to the number of frames in the dataset, and with a constant amount of memory that is independent of the number of views and length of the videos.
- An approximation to this method that transfers the effect of the image filter on a portion of the dataset to achieve significant speed-ups with only minor degradation of output quality.
- A set of modifications to PatchMatch that enables wide-baseline matching without quantization effects
- An interleaved approach for producing temporally stable results for iterative NPR filters such as neural style transfer

Our approach enables content creators to employ common image filters such as dehazing, auto color balance, stylization, and intrinsic decomposition to many camera array configurations for which these operations were previously unsuitable.

2. Spatio-temporal Consistency

We consider a dataset with several *views* $\{V_i\}$, e.g., a stereo dataset has two views V_1 and V_2, and a 5×5 light field has 25 views. We use x to denote the position of a pixel in a view; and for dynamic sequences, e.g., a stereo video, we use t to index the frames, i.e., V_{it} is the t^{th} frame of the i^{th} view. Our algorithm takes as input an unprocessed dataset $\{V_{it}\}$ and an image filter f. We name $\{P_{it}\} = \{f(V_{it})\}$ the dataset filtered frame-by-frame by f. We are interested in cases where f is not stable, i.e., the filtered dataset $\{P_{it}\}$ suffers from spatial and/or temporal inconsistencies. Our objective is to generate an output dataset $\{O_{it}\}$ that retains the visual appearance of $\{P_{it}\}$ while being spatially and temporally consistent.

Overview We design our algorithm to handle real world datasets with large numbers of views and arbitrary video lengths. To satisfy these requirements, we design our method to only consider a small number of views and frames at any given time because manipulating many frames and views together quickly become unwieldy due to the sheer size of the data. We proceed in two main steps. First, we select an *anchor view* denoted by a, and stabilize it. Second, we operate on the other views $i \neq a$, processing frames one at a time in temporal order (Fig. 2). One major advantage of our algorithm is that when we treat the frame P_{it}, we only need to consider its input counterpart V_{it}, its predecessor $V_{i(t-1)}$, and the corresponding output anchor frame O_{at}, resulting in little memory overhead. We formulate our problem as a least-squares optimization that we minimize with an off-the-shelf linear solver.

2.1. Stabilizing the Anchor View

The first step of our algorithm is to select by hand an anchor view. In principle this could be any view, but as we wish to maximise potential coverage in correspondence, then this is typically the most central view in the dataset. We name a its index. $\{V_{at}\}$, $\{P_{at}\}$, and $\{O_{at}\}$ are datasets restricted to a single view. That is, they are standard videos and we can use existing video techniques to remove any inconsistency introduced in $\{P_{at}\}$ by the image filter. For our prototype, we use the algorithm by Bonneel et al. [BTS*15] and

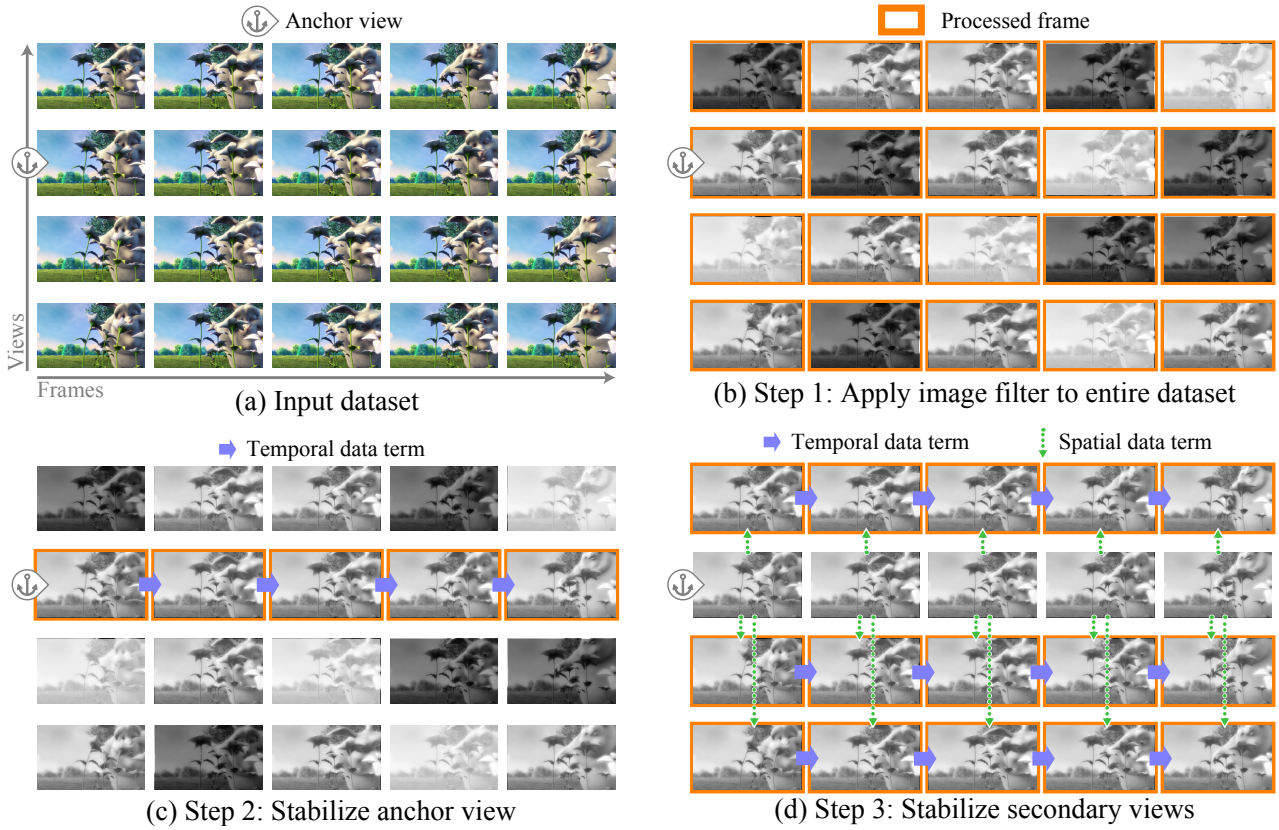


Figure 2: ***Algorithm overview.*** *We apply image filer f to input data $\{V_{ij}\}$ (a) to produce filtered data $\{P_{ij}\}$. We are interested in the case where these data suffer from spatial and temporal instability (b). First, we stabilize the anchor view $\{P_{at}\}$ using an existing technique to remove temporal inconsistencies from videos (c). Then, we stabilize the secondary frames $\{P_{ij}\}$ (for $i \neq a$) using the anchor frames $\{O_{at}\}$ as reference in addition to the previous frames (d). In Section 2.4, we reorder this computation to be more storage efficient without affecting the output: in practice, (c) and (d) are interleaved to create a streaming algorithm. Please view in color. ('Big Buck Bunny Flower': $\lambda_t = 0.1, \lambda_s = 0.1$.)*

its available implementation, although any other technique which removes temporal inconsistencies could be used. For completeness, we summarize this algorithm below.

Enforcing the temporal consistency of the filtered video $\{P_{at}\}$ is formulated as a least-squares optimization that seeks to generate an output $\{O_{at}\}$ with gradients similar to $\{P_{at}\}$ while minimizing the color variations along the time dimension, i.e., successive frames should look the same. The formulation is causal, that is, frames are processed one by one in temporal order and the result at a given frame depends only on past results. Furthermore, the result at frame $t > 0$ only depends on data at frames t and $(t-1)$. The least-squares energy to compute O_{at} is:

$$\arg\min_{O_{at}} \int \|\nabla O_{at} - \nabla P_{at}\|^2 + w_t(x)\|O_{at} - \mathcal{T}_{at}(O_{a(t-1)})\|^2 \mathrm{d}x \quad (1a)$$

$$\text{with: } w_t(x) = \lambda_t \exp(-\alpha_t \|V_{at} - \mathcal{T}_{at}(V_{a(t-1)})\|^2) \quad (1b)$$

where $\mathcal{T}_{at}(\cdot)$ is the temporal warp operator that places into correspondence the pixels at frame $(t-1)$ with those at t, and α_t and λ_t control how strongly temporal consistency is enforced. In practice, we adapt the PatchMatch algorithm [BSFG09] (§2.2) to compute a correspondence $\mathcal{T}_{at}$ between the input frames $V_{a(t-1)}$ and V_{at}.

This optimization problem is known as a Screened Poisson Equation. Finding a minimizer amounts to solving a sparse linear system, which can be completed with a standard linear solver. The result is a stabilized version $\{O_{at}\}$ of the anchor view. In the next section, we explain how we build upon this approach to process the other views.

2.2. Stabilizing the Secondary Views

Once we have stabilized the anchor view a, we process secondary views $i \neq a$ one by one. For each view, the process is causal and only involves data at t and $(t-1)$ similarly to Equation 1. The difference is the addition of a data term that relates the considered frame P_{it} to its corresponding stabilized anchor frame O_{at}:

$$\arg\min_{O_{it}} \int \|\nabla O_{it} - \nabla P_{it}\|^2 + w_t(x)\|O_{it} - \mathcal{T}_{it}(O_{i(t-1)})\|^2 + w_s(x)\|O_{it} - \mathcal{S}_{it}(O_{at})\|^2 \mathrm{d}x \quad (2a)$$

$$\text{with: } w_t(x) = \lambda_t \exp(-\alpha_t d(V_{it}, \mathcal{T}_{it}(V_{i(t-1)}))^2) \quad (2b)$$

$$w_s(x) = \lambda_s \exp(-\alpha_s d(V_{it}, \mathcal{S}_{it}(V_{at}))^2) \quad (2c)$$

where $\mathcal{S}_{it}(\cdot)$ is the spatial warp operator that puts in correspondence the pixels in view i with those in the anchor view a, $d(.,.)^2$ denotes the quality of alignment, computed as the sum of squared pixel difference over a 7×7 patch neighborhood and α_s and λ_s are parameters that controls the influence of the view consistency term. We use a modified version of PatchMatch to define the operators $\mathcal{S}_{it}$ and $\mathcal{T}_{it}$ (§2.3). In this formulation, λ_t controls the regularization strength over time and λ_s across views, while α_t and α_s control the confidence we give to the computed correspondences. In practice, we use $\lambda_t \in [1,50]$, $\lambda_s \in [0.2,10]$ and $\alpha_t = \alpha_s \in 7 \times 7 \times [0.1,1]$. We found that increasing α_t and α_s resolves flickering that could occur at object boundaries due to poor correspondences. Inferring temporal consistency only from the anchor view (that is, $\lambda_t = 0$ for $i \neq a$) did not produce satisfactory results (see accompanying video).

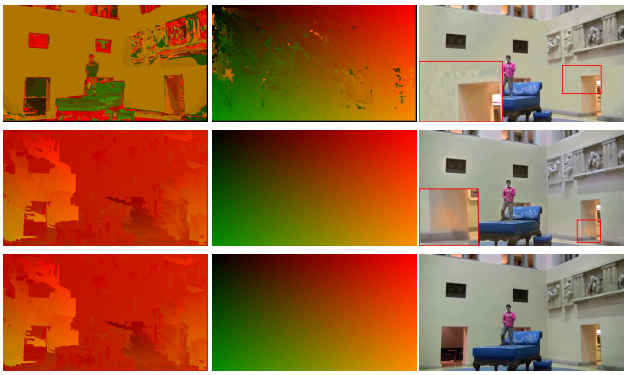
Akin to Equation 1, this process is a least-squares optimization problem that amounts to solving a Screened Poisson Equation. The sparsity of the system to solve is the same since the new spatial term $\|O_{it} - \mathcal{S}_{it}(O_{at})\|^2$ only changes the data-attachment term that was already nonzero due to the temporal term. Memory use is higher since the energy function relies on 7 frames instead of 5, but this is well within the capacity of current hardware. And most importantly, the memory requirement of our approach does not dependent on the number of views and frames in a dataset, e.g., we can process arbitrarily long sequences within a fixed amount of memory. The computational cost is also higher because we run PatchMatch twice (for $\mathcal{S}$ and $\mathcal{T}$), but practically PatchMatch is cheap. As the process at each frame amounts to running PatchMatch at most twice and solving a linear system of the same size and sparsity that is independent of the number of views and frames, the computational complexity of processing a dataset is linear in the number of views and frames, that is: $\mathcal{O}(N_v N_f)$, where N_v is the number of views in a dataset and N_f is the number of frames per view.

Fourier Analysis Bonneel et al. [BTS*15] used Fourier analysis to study the frequency content of the solution of Equation 1. They showed that most of the low frequencies come from the previous output frame $O_{a(t-1)}$ and that the processed frame P_{at} has more influence on the high frequencies. In the appendix, we derive a similar result for our case (Eq. 2) and show that while the high frequencies still mostly come from the processed frame P_{it}, the low frequencies are dominated by a blend of the previous frame $O_{i(t-1)}$ and of the anchor frame O_{at}, the balance between them being controlled by w_s and w_t.

2.3. Spatio-temporal correspondences for $\mathcal{T}$

An important design decision is the choice of the warp operator $\mathcal{T}$ in Eqs. 1 and 2. There is an extensive body of work on establishing correspondences between images. For this, our requirements are twofold: we seek a technique that is fast enough to scale to our large datasets, and that can cope with large viewpoint changes present in wide-baseline datasets. We begin by describing initial experiments which provided insights to our final approach.

Sparse correspondences can be computed by matching across wide baselines and time with features like SIFT [Low04], refined through motion models via RANSAC. In practice, sparse correspondences are insufficient to remove inconsistency as many filters introduce high-frequency spatial variations. For example, intrinsic decomposition creates different gradient distributions in different regions from frame to frame. Sparse correspondences can only remove low spatial frequency deviations because the spatial resolution of the correspondences does not match the flickering resolution.



(a) Corresp. to anchor (b) Corresp. in time (c) Output effect

Figure 4: *Incorrect correspondence effect (a,b) on regularized output (c), 'Magician'.* Row 1: *The original PatchMatch algorithm occasionally produces repeated patches (a) and temporally inconsistent flow (b), leading to temporally-unstable quantization effect (c).* Row 2: *We handle correspondence inaccuracies in wide baselines via several alterations of the original algorithm (§2.2). This effectively reduces quantization artifacts.* Row 3: *With the same correspondence, controlling the diffusion process with the inter-patch distance prefers good correspondences and improves the result.*

Optical flow techniques are one way to provide dense per-pixel correspondences, and these commonly trade accuracy for efficiency. However, most are designed for small image changes in time and so enforce strong spatial regularization. This may be suitable for light field camera arrays, but is not suitable for wider baselines. For instance, PCA Flow [WB15] is efficient but does not produce accurate correspondences for wide-baseline camera arrays as the spatial regularization is too strong. Our supplemental materials [sup15] illustrate the artifacts produced by PCA Flow on wide-baseline sequences. Large motion optical flow techniques relax this constraint (e.g., EpicFlow [RWHS15]), but can be prohibitively slow and so do not scale to our problem.

Nearest neighbor field techniques solve for matches that are good purely in terms of patch appearance, and this allows them to find correspondences across wide-baseline views that have significant differences. PatchMatch is one efficient approach which uses random sampling; however, it can produce correspondence fields that are not geometrically meaningful or temporally consistent. In many cases, this is not a problem, particularly when the frame difference in time or view is small such that the matching patch distance is also small. However, large frame difference can lead to inconsistent 'quantization'-style artifacts where a single source patch acts as correspondence for many target patches (Fig. 4).

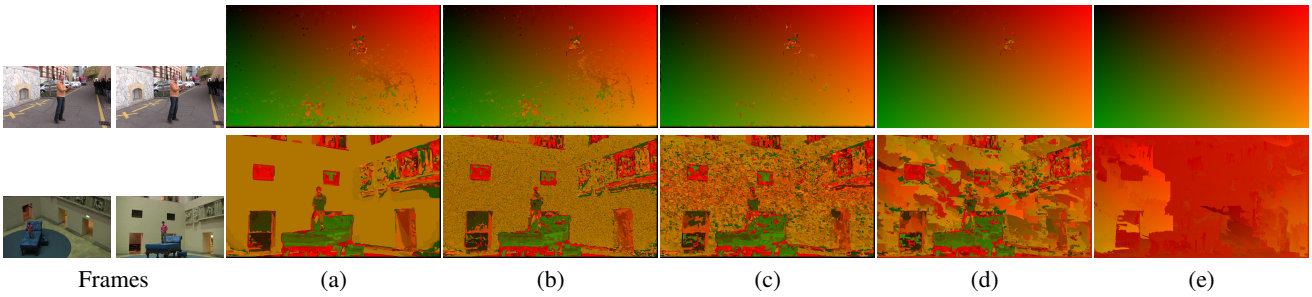


Figure 3: *Correspondences in time (first row, 'Juggler') and across views (second row, 'Magician'). (a) The original PatchMatch algorithm can lead to many small and repeated patches, which occasionally causes quantization artifacts in the output. (b) Limiting the same patch to be the source for no more than 3 correspondences helps vary the patch source. (c) In areas of uniform luminance, we do not perform the PatchMatch random search, which reduces random errors. (d) Combined with a rough motion estimate, patches become temporally consistent. (e) Finally, we restrict the patch search to a small radius around the current estimate as offsets over time are unlikely to change drastically. Please zoom.*

Our solution For these cases, we combine both approaches and constrain PatchMatch. First, we compute sparse SIFT features and use them to fit similarity motion model using RANSAC with a large error threshold (200 pixels). The resulting motion field is globally smooth but locally inaccurate because of parallax and non-rigid motion. Thus, second, we initialize PatchMatch with this motion field and perform randomized refinement. To prevent PatchMatch from moving too far from the initialization: a) we limit the search window over which the random search step in PM searches to 20 pixels, b) we explicitly enforce a bijectivity constraint on the correspondences by limiting one source patch to match at most three destination patches, to avoid one-to-many quantization artifacts, and c) we do not perform the random search step on patches with luminance standard deviation smaller than 10 (on a scale of $[0,255]$), as these low-contrast correspondences are often arbitrary and erroneous. Finally, we use the PatchMatch patch appearance error—the sum of squared differences over the patch—to weight the strength of the correspondence. This is more robust than the noise-sensitive pixel-wise weights used by Bonneel et al. [BTS*15]. Figure 3 shows how each of these changes result in a flow field that is more smooth and meaningful, and also more consistent over time, which leads to visually pleasing results (Fig. 4).

2.4. Efficient Computation Reordering

For the sake of clarity, we have described the algorithm as (1) create the filtered dataset $\{P_{it} = f(V_{it})\}$, (2) process the anchor view, (3) process the secondary views (Fig. 2). We can improve this scheme by reordering the operations in two ways that do not affect the result. First, we observe that $P_{it} = f(V_{it})$ is only needed when computing the output frame O_{it}. Instead of applying the image filter f to the entire dataset and caching the result, it is more efficient to compute $f(V_{it})$ when we start the processing of P_{it} and discard it at the end of it. It does not change the result but removes the need to cache the entire $\{P_{it}\}$ dataset. Second, the computation of anchor and secondary views depends only on data at times $(t-1)$ and t, and the secondary views depend on the anchor view but not the opposite. This allows us to interleave the processing of the anchor and secondary views so that the entire process is causal. That is, assuming that we have processed all the data up to $(t-1)$ (for $t>0$), we first compute the output anchor view O_{at} with Equation 1, and use the result to compute the output secondary views O_{it} (for $i \neq a$) with Equation 2. This allows us to store the data older than $(t-1)$ directly in a compressed format, e.g., H264 or H265. Since this data is not used in the computation anymore, the loss introduced by the compression has no impact on the output quality.

2.5. Speed-up via Filter Transfer

The approach described produces stable outputs for many datasets and filters (§3). However, applying image processing filters across multiple frames and views can be prohibitively expensive when the image filter f is costly. We propose an approximation which enables a significant speed-up to the per-frame filtering while maintaining a satisfying output. Our strategy is to filter only a subset of the frames, i.e., we compute $f(V_{it})$ only for some i and t values and let our regularization technique transfer the response from these frames (Fig. 5). With this strategy, we exploit the high level of redundancy between nearby frames and views to propagate the effect of the image filter f from a few filtered frames onto all the others even if they have not been filtered. In our algorithm, the anchor frames are more important because they regularize the secondary frames. Building on this observation, we filter all the anchor frames as normal, but filter only one in every n secondary frames. Because our approach is causal, we also filter the first frames of all views, i.e., we compute $f(V_{i0})$ for all i. Formally, we define the dataset $\{\tilde{P}_{it}\}$:

$$\tilde{P}_{it} = \begin{cases} f(V_{it}) & \text{if } (i=a) \text{ or } (t \bmod n \equiv 0) \\ V_{it} & \text{otherwise} \end{cases} \quad (3)$$

Then, the algorithm is the same, i.e., we minimize Equations 1 and 2 using $\tilde{P}$ instead of P. As we filter all anchor frames, Equation 1 is unchanged; only secondary view processing (Eq. 2) is affected.

The value of keeping the input frame V_{it} in place when $(i \neq a)$ and $(t \bmod n \not\equiv 0)$ is to maintain high-frequency detail in the output. Consider replacing V_{it} with a mid-gray frame instead. Our consistency term propagates lower-frequency content, but the high frequencies are dominated by the current frame so that scene dynam-

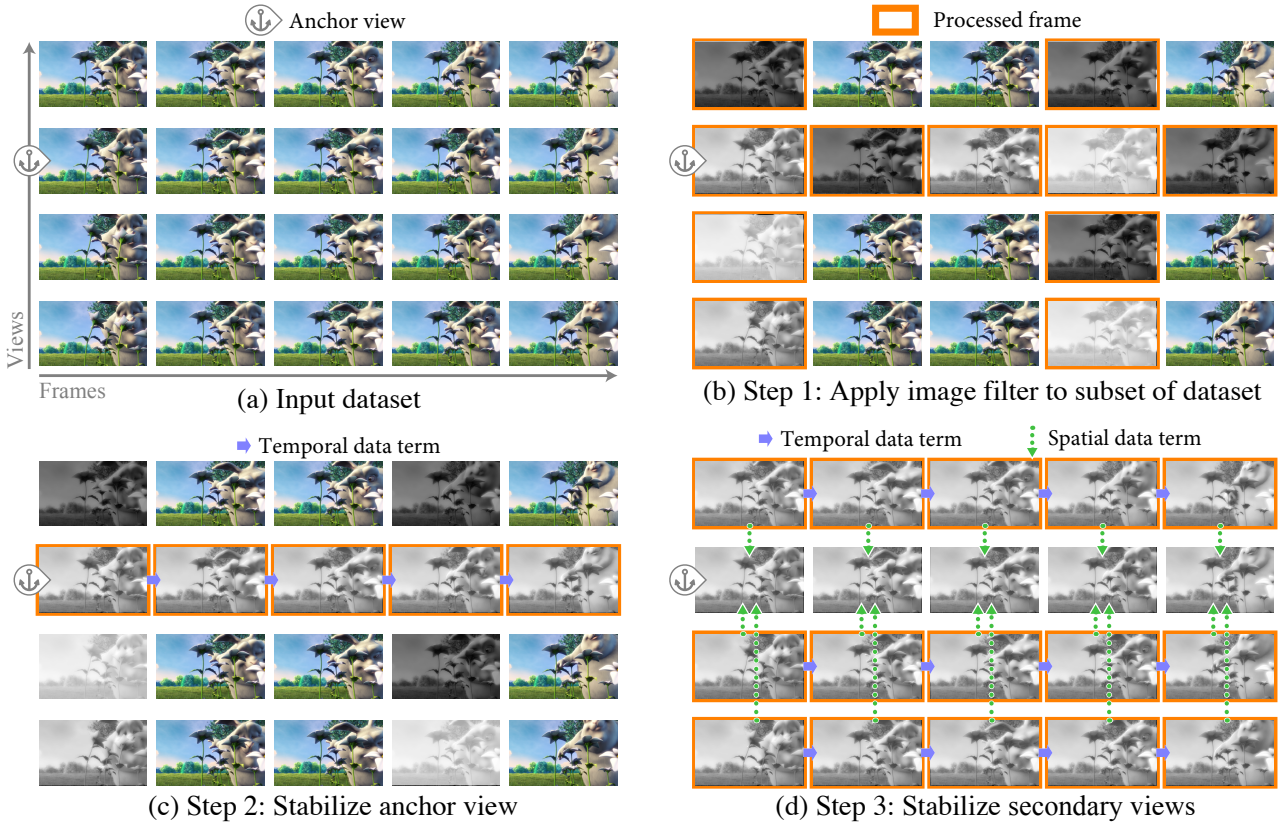


Figure 5: *We accelerate our algorithm by applying image filter f only to a subset of the input data (a). We filter all frames at $t = 0$ and all anchor frames, and 1 in n frames for the secondary views (b). We use $n = 3$ in this example. The rest of the algorithm remains the same (c,d). This significantly speeds up the process while having a limited impact on the results because our algorithm is sufficiently robust to transfer the information from the filtered frames, essentially "hallucinating" the effect of the filter. We reorder this computation to be more storage efficient by interleaving (c) and (d) to create a streaming algorithm (§2.4). Please view in color. ('Big Buck Bunny Flower': $\lambda_t = 0.1, \lambda_s = 0.1$.)*

ics are not lost (see Appendix). Thus, inserting a mid-gray frame would blur the result when $(i \neq a)$ and $(t \bmod n \not\equiv 0)$. Using V_{it} as a proxy for high-frequency content relies on the fact that, for many filters, the fine gradients are not altered significantly between the input and processed videos (e.g., not flipped). This condition is also present in reverse: a filter which removes detail, and is then subset with V_{it} as a proxy, will see detail bleed back in.

However, for many filters these conditions are maintained and the result is satisfactory. We found that speed-up factors of $5\times$ were possible while keeping a satisfying output quality. Figure 6 quantifies the output quality as we transfer more filtered frames.

3. Results

We found that our approach performs well on a broad range of image filters as shown in our experiments. First, we report run times and memory consumption, then we run experiments with a variety of filters, scenes, and camera array configurations. We encourage the reader to watch the supplemental videos [sup15].

Performance Our algorithm is efficient, and scales *linearly* with the number of views. In many cases, camera array sequences either have high angular and low spatial resolution, or high spatial and low angular resolution. Our algorithm scales linearly with angular resolution, and super-linearly with spatial resolution due to our multi-scale Poisson solver and PatchMatch. On a 6-core Intel Xeon 3.5 GHz, a low resolution 320×240 15-view sequence is regularized at a rate of approximately 2.7 seconds per frame for all views, while a high resolution 1280×768 91-view sequence takes roughly 2 minutes per frame for all views, or 1.3 seconds per image. With our unoptimized implementation, this corresponds to a throughput of 0.8 mega-pixel per second. In practice, 60% of our runtime is spent in computing the PatchMatch correspondence field and SIFT matches. Currently, for convenience, our prototype implementation holds two frames per camera in memory, along with correspondence fields and other associated data. As such, the total memory usage is $\approx$6 GB for a 1280×768 91-view sequence independently of the sequence length. However, in principle, it is not necessary to store more than seven frames in memory at once (§2.2) and we could optimize our implementation to further reduce its memory footprint.

Color mapping Automatic color adjustment filters are common, and so we tested both Adobe Photoshop and Premiere Pro on various multi-camera setups: light fields ('Truck', 'Aquarium'), stereo

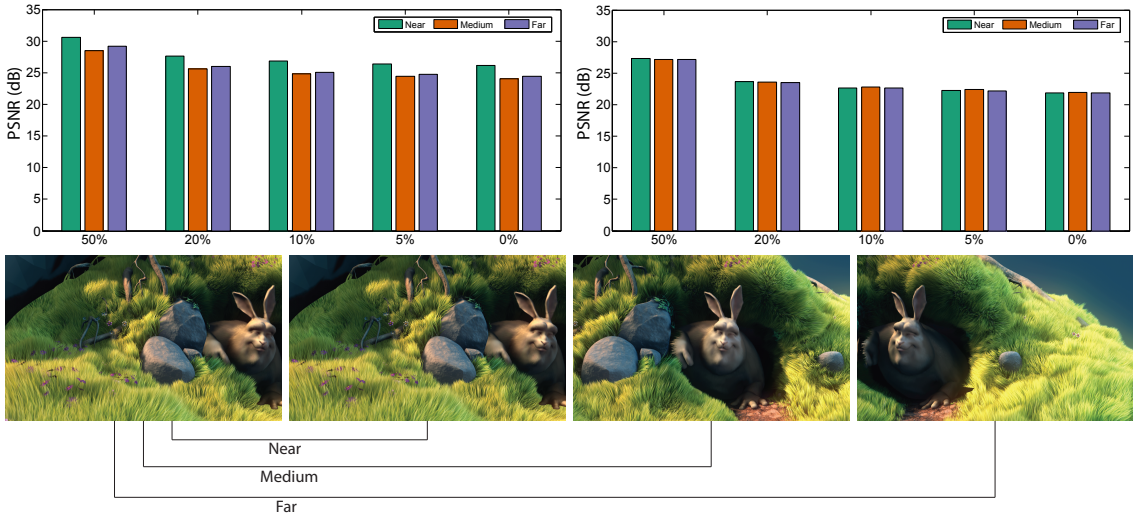


Figure 6: *We evaluate the error induced by our filter transfer strategy between an anchor and a secondary view separated by either near, medium, or far distances. The secondary view intrinsic decomposition reflectance filtering has been subsampled by factors of 2, 5, 10, 20, and entirely except for its first frame. The anchor view has been either fully processed (*left*) or subsampled similarly to the secondary view (*right*). Processing the anchor view entirely improves the reconstruction quality of the secondary view. ('Big Buck Bunny Rabbit':* $\lambda_t = 0.2, \lambda_s = 0.2$*.)*

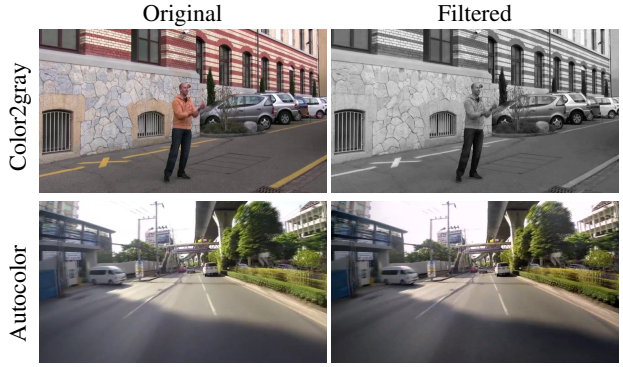


Figure 7: *Two examples of other filters that resulted in flickering that was corrected. ('Juggler':* $\lambda_t = 0.2, \lambda_s = 0.2$*. 'Bangkok Chaos':* $(\lambda_t = 0.01, \lambda_s = 0.01.)$*)*

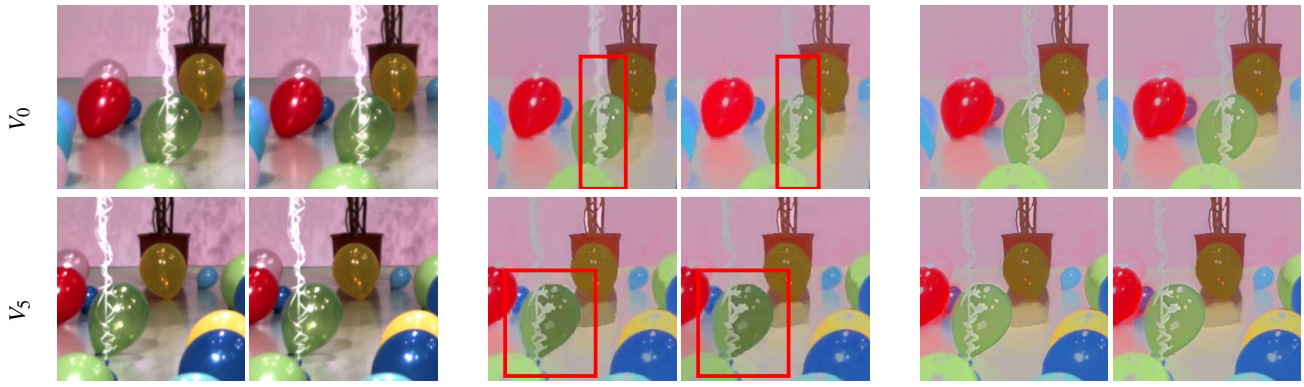
videos ('Bangkok Chaos'), and multi-view setups ('Magician'). Common artifacts are flickering, which is often caused by bright object, such as the sun or a flame, entering the frame, and more subtle color casts which vary between views and over time as occlusions reveal unseen areas. Smoothing AutoColor parameters over video sequences can remove temporal flicker, but fails to remove color cast inconsistencies between views. In most cases, our approach is able to correct for both artifacts. However, the lack of correspondences between views in the 'Magician' sequence leaves some temporal inconsistencies. Similar flickering artifacts are seen with other color mapping functions such as tone mapping and contrast-preserving decoloration (color2gray), which we successfully remove.

Stylization Stylization methods often perform some kind of color mapping in combination with edge highlights, which can exhibit flickering over time and space. We show a panoramic example using the watercolor filter from Adobe Photoshop applied to each frame, and then regularized. In general, filters which add new edges which are inconsistent with input edges will lead to those new edges being smoothed out during consistency enforcement (Fig. 7).

Dehazing We ran the Photoshop Dehazing filter on each frame of a stereo video, which estimates an atmospheric haze color from image statistics, and uses this to remove haze in outdoor images. In this case, small changes in this estimate led to drastic changes in the result, which we were able to make consistent (Fig. 1).

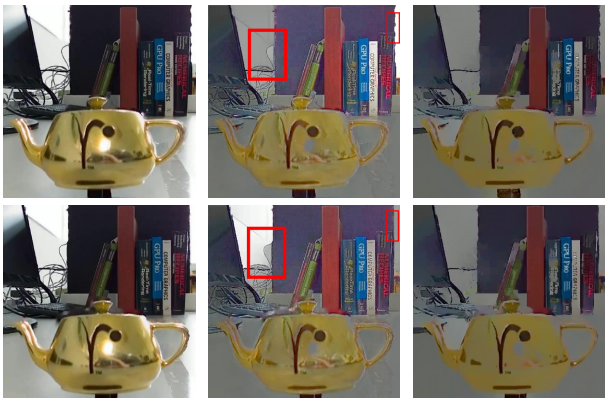
Intrinsic Decomposition Intrinsic decomposition is an essential step in reflectance or lighting editing, as well as for analyzing scene compositions. We computed a per-frame decomposition into reflectance and albedo information using Bell et al. [BBS14] (Fig. 8). This filter shows significant flickering across time and view, but our approach is able to enforce consistency across both. In Fig. 9, we show a comparison with the recent (static) light field intrinsic decomposition method of Garces et al. [GEZ*16]. Additional comparisons can be found in supplemental materials.

Non-photorealistic Rendering To help evaluate which filters are appropriate for our approach, we also consider NPR filters which create a significantly pronounced effect. As our algorithm works in the gradient domain, it implicitly assumes that the input edge structure can effectively regularize the filter. Thus, our approach is not appropriate for filters which add significant new edges to the output, e.g., Adobe Photoshop cutout filter (Fig. 12). This is because when there is a new edge in the output that is missing from the input, our approach blurs it away, producing an unsatisfying result.



(a) *Input Videos (cropped)* **(b)** *Per-frame intrinsic albedo* **(c)** *Our result after enforcing consistency*

Figure 8: *Decomposing light field videos into albedo and reflectance information produces unstable results due to changes in content over time and view. For instance, in (b), V_0, the balloon string brightness changes over time; in V_5, the green balloon has different appearance either side of the string. Our method greatly reduces the resulting flicker, yielding a consistent result. ('Balloon': $\lambda_t = 0.2, \lambda_s = 0.02$.)*



(a) Input Images (b) [GEZ*16] (c) Our result

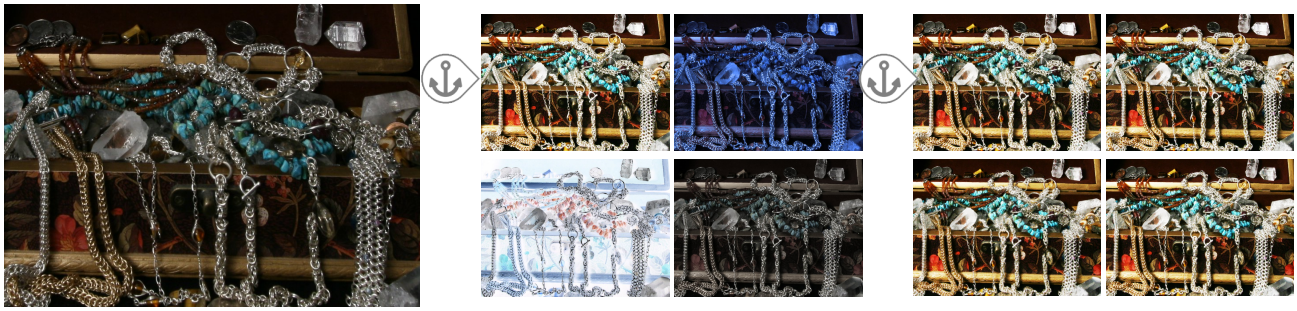
Figure 9: *We compare our approach with that of Garces et al. [GEZ*16] which operates on static light field images. We enforce consistency across views on the albedo obtained via the method of Zhao et al. [ZTD*12]. The method of Garces et al. exhibits consistency artifacts across views, shown in red. ('Teapot': $\lambda_t = 0, \lambda_s = 1$.)*

Figure 10: *Iterative NPR filters can benefit from our regularization. We stabilize the Gatys et al. approach [GEB15]. Our process tends to average out high frequencies over time; however, integrating it into the NPR loop reintroduces high frequencies and maintains the desired style. ('Big Buck Bunny Flower': $\lambda_t = 0.1, \lambda_s = 0.1$.)*

However, for iterative filters that introduce new edges, such as the popular neural style transfer [GEB15] or 'deep dream' effects [MOT15], we propose an iterated (filter → regularize → filter) approach to cope with this edge blurring problem.By interleaving our regularization within style transfer iterations, we are able to remove significant inconsistency at low computational cost, while retaining the desired style, as the high frequencies are added back at each iteration. We perform this iteration four times to create a stable result, as shown in the video and in Fig .10. One contemporaneous approach for temporally stable style transfer uses long-term features to track objects across motions in videos [RDB16]. While effective, this approach relies on optical flow and takes roughly 3 minutes per image pair, which would be prohibitive when scaling to large numbers of views. Further, their approach modifies the underlying formulation [GEB15], while ours is agnostic to the formulation and so applies to a class of problems.

Random Filters To demonstrate the efficacy of our filter transfer approach, we applied random color transforms to all but the anchor view of the 'Treasure' static light field (Fig. 11). Despite the drastic frame-to-frame differences, our approach is still able to produce a mostly consistent output. While a real world per-frame filter is unlikely to produce such extreme variations, this test shows that when the gradients are kept intact, our approach can still stabilize the views in the presence of significant instability.

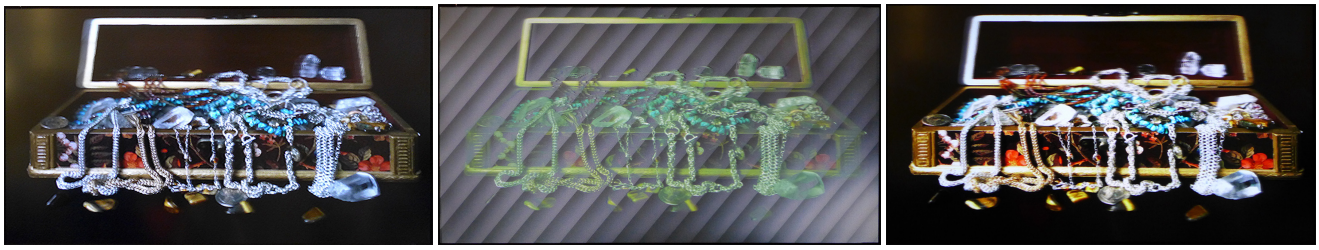
Validation We evaluate our method against existing per-view approaches to produce temporally stable output [LWA*12, BTS*15], as well as to current solutions in Adobe Premiere Pro. To demonstrate the improvements in our supplemental video, we 1) express



(a) *One view of a 16-camera light field.*

(b) *Random filters applied to each view.*

(c) *The same views after our approach.*



(d) *Output on an automultiscopic display (left to right: input, random filters causing view artifacts, our output).*

Figure 11: *Torture test: We chose a random filter effect for each view (b). The anchor frame (top left) stabilizes the remaining views. Even with vastly different appearances, our approach still creates a consistent output (c). When viewed on an automultiscopic display (d), view inconsistencies cause spatially-varying artifacts which change over views. Our approach removes them. ('Treasure': $\lambda_t = 0, \lambda_s = 10$.)*

view and time changes by pausing time and switching views, and 2) recorded an auto-multiscopic display with real camera motion.

Additionally, we also compare our approach to the single view method of Bonneel et al. [BTS*15] but with a naïve traversal of the light field: generating a single video which sweeps through all views of the multi-view sequence in one direction, then advances one frame in time, then sweeps through all views in the other direction, then advances one frame, and so on. While better than a pure per-view approach, the sweeps leave artifacts between distant views over our anchor approach, and causes 'leaking' of the input when using filter transfer speed-up (please see our supplemental material [sup15]).

4. Discussion

We typically choose the anchor view to be in the middle of the camera arrays to limit the distance of the farthest view. This choice works well in our experiments and we observed that distance has only a limited impact on accuracy (Fig. 6). However, manufacturers have recently announced spherical camera arrays covering 360° and distance might become a concern with the datasets that they produce. If that is the case, extending our approach to several anchors will be an interesting direction of future work.

To propagate the information spatially, we experimented with referring to the nearest view instead of the anchor but this did not perform as well. Distant views suffered from error accumulation because they were related to the anchor through a chain of intermediate views, and temporal inconsistencies in the correspondence field were thus amplified for these views. Direct anchor attachment does not have an accumulation effect and performed better in our case, however, investigating multiple anchor views and loop closure to prevent drift is an interesting area for future work. While we experimented with a fixed subsample factor to speed up the per-frame processing, a prior video analysis would allow automatic subsample factor adjustment based on motion or content. Our work favored a solution for which no precomputation is necessary.

Input Our result

Figure 12: *Limitation: NPR filters which add strong gradients in regions where none exist in the input image, e.g., Photoshop Cutout, can lead to blurring effects after the video is made consistent. ('Kendo': $\lambda_t = 0.1, \lambda_s = 0.1$.)*

Parameters Only two parameters need to be changed between scenes, and these changes are generally intuitive. Scenes with wider baselines require stronger spatial regularization, and scenes with fast motion (blur) require stronger temporal regularization. Changes

in scene depth across time are regularized by the input gradients, and do not typically require parameter changes. Please see the figure captions for the parameters used.

Correspondence One might ask why we do not use depth maps or depth-aware correspondence fields [HRF13]. Relying on image-based rather than strict geometric correspondences provides flexibility and relaxes input requirements, e.g., enabling our handheld wide baseline street juggler example. This argument is corroborated by works on generating light field imagery [RPZSH13, AGB*16], which also use image correspondences (optical flow) rather than depth, even with known camera configurations. Less strict correspondence also allow robustness to view-based effects (specularity, depth of field) and rolling shutter issues.

Standard PatchMatch introduces posterization artifacts, but by adding rigid motion initialization, limiting the search radius and number of repeated target patches, and skipping constant source patches, we effectively increase robustness and reduce artifacts. However, in hard cases, a little randomness *adds* robustness: in wide baseline cases, for some image regions, there is no good match between views. Retaining some of this randomness allows our approach to still find similar looking spatio-temporally stable regions for consistent filter transfer. The same rationale extends to comparisons against depth-based reconstruction methods, where strict geometric reasoning will fail but a more flexible pseudo-random approach will be more successful. Finally, our image-based correspondences are still only soft constraints which guide spatio-temporal consistency in conjunction with the input video gradients. This provides further flexibility.

That said, some sequences can be challenging, e.g., 'Magician', where artifacts in shadows and smooth gradients come from correspondence errors. While our method helps reduce 'one-to-many' correspondence errors over standard PatchMatch in wide-baseline or low-textured sequences, some sequences still produce artifacts due to the challenge of finding correspondence in general situations where disocclusion between views may be large. Prior information about the camera layout, or improvements in registration, will help generalize our approach.

Conclusion

We have introduced a technique that enables the application of unstable image filters such as dehazing, autocolor, and intrinsic decomposition to datasets recorded with camera arrays, including stereo videos, and static and dynamic light fields. Without our algorithm, these filters introduce temporal flickering and view inconsistencies that render their output unusable. We have shown how to remove these artifacts by solving a least-squares optimization problem that has the property of scaling linearly with the number of cameras in the array and with the duration of sequence, and of running within a fixed amount of memory independent of these parameters. For computationally expensive image filters, we have proposed an acceleration strategy that runs the filter only a subset of the data and then transfers the result to nearby frames and views. Our experiments have demonstrated that our approach enables a large class of standard filtering operations on media like stereo and light field videos that are rapidly gaining in popularity.

Acknowledgements

We thank Kovács et al. [KFLA15], Ballan et al. [BBPP10], the (New) Stanford Light Field Archive, Nagoya University Multi-view Sequence Download List, Al Caudullo Productions, and G. Pouillot for their videos. We thank Szo-Po Wang and Wojciech Matusik for use of their auto-multiscopic display, and Serena Booth for her narration. For this work, James Tompkin and Hanspeter Pfister were sponsored by the Air Force Research Laboratory and DARPA Memex program. Nicolas Bonneel thanks Adobe for software donations.

References

[AG16] ALPEROVICH A., GOLDLUECKE B.: A variational model for intrinsic light field decomposition. In *Asian Conference on Computer Vision (ACCV)* (2016). 2

[AGB*16] ANDERSON R., GALLUP D., BARRON J. T., KONTKANEN J., SNAVELY N., HERNÁNDEZ C., AGARWAL S., SEITZ S. M.: Jump: Virtual reality video. *ACM Trans. Graph. 35*, 6 (Nov. 2016), 198:1–198:13. 10

[AZJ*15] AO H., ZHANG Y., JARABO A., MASIA B., LIU YEBIN L., GUTIERREZ, QIONGHAI D.: Light field editing based on reparameterization. In *Adv. in Multimedia Information Proc.* (2015). 2

[BBPP10] BALLAN L., BROSTOW G. J., PUWEIN J., POLLEFEYS M.: Unstructured video-based rendering: Interactive exploration of casually captured videos. 1–11. 10

[BBS14] BELL S., BALA K., SNAVELY N.: Intrinsic images in the wild. *ACM Trans. on Graphics (SIGGRAPH) 33*, 4 (2014). 7

[BPK*13] BAEK J., PAJAK D., KIM K., PULLI K., LEVOY M.: WYSIWYG computational photography via viewfinder editing. *ACM Trans. Graph. 32*, 6 (2013). 2

[BSFG09] BARNES C., SHECHTMAN E., FINKELSTEIN A., GOLDMAN D. B.: PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph. (SIGGRAPH) 28*, 3 (2009). 2, 3

[BTS*15] BONNEEL N., TOMPKIN J., SUNKAVALLI K., SUN D., PARIS S., PFISTER H.: Blind video temporal consistency. *ACM Trans. Graph. (SIGGRAPH Asia) 34*, 6 (2015). https://github.com/nbonneel/blindconsistency. 1, 2, 4, 5, 8, 9, 11

[BZCC10] BHAT P., ZITNICK C. L., COHEN M., CURLESS B.: GradientShop: A gradient-domain optimization framework for image and video filtering. *ACM Trans Graph (SIGGRAPH) 29*, 2 (2010). 2

[DD10] DELON J., DESOLNEUX A.: Stabilization of flicker-like effects in image sequences through local contrast correction. *SIAM Journal on Imaging Sciences 3*, 4 (2010), 703–734. 2

[DYY15] DANG K., YANG J., YUAN J.: Adaptive exponential smoothing for online filtering of pixel prediction maps. In *2015 IEEE International Conference on Computer Vision (ICCV)* (Dec 2015), pp. 3209–3217. 2

[FK00] FOOTE J., KIMBER D.: Flycam: Practical panoramic video and automatic camera control. In *IEEE Int. Conf. on Multimedia and Expo (ICME)* (2000), vol. 3, pp. 1419–1422. 1

[FL11] FARBMAN Z., LISCHINSKI D.: Tonal stabilization of video. *ACM Trans. on Graphics (SIGGRAPH) 30*, 4 (2011), 89:1 – 89:9. 2

[GEB15] GATYS L. A., ECKER A. S., BETHGE M.: A neural algorithm of artistic style. *CoRR abs/1508.06576* (2015). 8

[GEZ*16] GARCES E., ECHEVARRIA J. I., ZHANG W., WU H., ZHOU K., GUTIERREZ D.: Intrinsic light fields. *CoRR abs/1608.04342* (2016). 2, 7, 8

[HRF13] HERBST E., REN X., FOX D.: Rgb-d flow: Dense 3-d motion estimation using color and depth. In *ICRA* (2013), IEEE. 10

[HSGL13] HACOHEN Y., SHECHTMAN E., GOLDMAN D. B., LISCHINSKI D.: Optimizing color consistency in photo collections. *ACM Trans. Graph. (SIGGRAPH) 32*, 4 (2013), 85:1 – 85:9. 2

[JMB*14] JARABO A., MASIA B., BOUSSEAU A., PELLACINI F., GUTIERREZ D.: How do people edit light fields? *ACM Trans. Graph. (SIGGRAPH) 33*, 4 (2014). 2

[JMG11] JARABO A., MASIA B., GUTIERREZ D.: Efficient propagation of light field edits. In *Proc. of the V Ibero-American Symposium in Computer Graphics* (2011), SIACG 2011, pp. 75–80. 2

[KFLA15] KOVÁCS P. T., FEKETE A., LACKNER K., ADHIKARLA V. K.: Big buck bunny light-field test sequences. MPEG contribution (ISO/IEC JTC1/SC29/WG11 M35721), Feb. 2015, 2015. 10

[LBP*12] LAFFONT P.-Y., BOUSSEAU A., PARIS S., DURAND F., DRETTAKIS G.: Coherent intrinsic images from photo collections. *ACM Trans. Graph. 31*, 6 (2012). 2

[LHW*10] LANG M., HORNUNG A., WANG O., POULAKOS S., SMOLIC A., GROSS M.: Nonlinear disparity mapping for stereoscopic 3d. *ACM Trans. Graph. 29*, 3 (2010), 10. 2

[LNJ13] LIU F., NIU Y., JIN H.: Joint subspace stabilization for stereoscopic video. In *Computer Vision (ICCV), 2013 IEEE International Conference on* (Dec 2013), pp. 73–80. 2

[Low04] LOWE D. G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60*, 2 (Nov. 2004), 91–110. 4

[LSS*15] LUO S.-J., SUN Y.-T., SHEN I.-C., CHEN B.-Y., CHUANG Y.-Y.: Geometrically consistent stereoscopic image editing using patch-based synthesis. *IEEE Trans. Vis. and Comp. Graph. 21*, 1 (2015), 56–67. 2

[LvBK*10] LO W.-Y., VAN BAAR J., KNAUS C., ZWICKER M., GROSS M.: Stereoscopic 3d copy & paste. *ACM Trans. Graph. 29*, 6 (2010), 147:1–147:10. 2

[LWA*12] LANG M., WANG O., AYDIN T., SMOLIC A., GROSS M.: Practical temporal consistency for image-based graphics applications. *ACM Trans. Graph. (SIGGRAPH) 31*, 4 (2012), 34:1–34:8. 2, 8

[MHCP12] MORSE B., HOWARD J., COHEN S., PRICE B.: Patchmatch-based content completion of stereo image pairs. In *Int. Conf 3D Imaging, Mod., Proc., Vis. and Transmission (3DIMPVT)* (2012), pp. 555–562. 2

[MJG14] MASIA B., JARABO A., GUTIERREZ D.: Favored workflows in light field editing. In *Int. Conf. on Comp. Graph., Vis., Computer Vision and Image Proc.* (2014), CGVCVIP. 2

[MOT15] MORDVINTSEV A., OLAH C., TYKA M.: Inceptionism: Going deeper into neural networks, June 2015. 8

[NLB*05] NG R., LEVOY M., BRÉDIF M., DUVAL G., HOROWITZ M., HANRAHAN P.: Light field photography with a hand-held plenoptic camera. *Computer Science Technical Report CSTR 2*, 11 (2005). 1

[PKCK06] PITIÉ F., KENT B., COLLIS B., KOKARAM A.: Localised deflicker of moving images. In *IEEE European Conference on Visual Media Production* (2006). 2

[PSZ*15] PERAZZI F., SORKINE-HORNUNG A., ZIMMER H., KAUFMANN P., WANG O., WATSON S., GROSS M. H.: Panoramic video from unstructured camera arrays. *Comp. Graph. Forum 34*, 2 (2015), 57–68. 1

[RAKRF08] RAV-ACHA A., KOHLI P., ROTHER C., FITZGIBBON A.: Unwrap mosaics: A new representation for video editing. *ACM Trans. Graph. 27*, 3 (2008). 2

[RDB16] RUDER M., DOSOVITSKIY A., BROX T.: Artistic style transfer for videos. 8

[RPZSH13] RICHARDT C., PRITCH Y., ZIMMER H., SORKINE-HORNUNG A.: Megastereo: Constructing high-resolution stereo panoramas. In *2013 IEEE Conference on Computer Vision and Pattern Recognition* (June 2013), pp. 1256–1263. 10

[RWHS15] REVAUD J., WEINZAEPFEL P., HARCHAOUI Z., SCHMID C.: EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *CVPR* (2015). 4

[SK02] SEITZ S. M., KUTULAKOS K. N.: Plenoptic image editing. *International Journal of Computer Vision 48*, 2 (2002). 2

[SSML14] SHENG B., SUN H., MAGNOR M., LI P.: Video colorization using parallel optimization in feature space. *IEEE Trans. Circuits and Systems for Video Tech. 24*, 3 (2014), 407–417. 2

[sup15] Supplemental materials for consistent video filtering for camera array. http://liris.cnrs.fr/~nbonneel/cameraarrays/data/, 2015. 4, 6, 9

[VLD*13] VENKATARAMAN K., LELESCU D., DUPARRÉ J., MCMAHON A., MOLINA G., CHATTERJEE P., MULLIS R., NAYAR S.: Picam: An ultra-thin high performance monolithic camera array. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 166:1–166:13. 1

[WB15] WULFF J., BLACK M. J.: Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *IEEE Conference on Computer Vision and Pattern Recognition* (June 2015). 4

[WJYG08] WANG L., JIN H., YANG R., GONG M.: Stereoscopic inpainting: Joint color and depth completion from stereo images. In *IEEE Conference on Computer Vision and Pattern Recognition* (2008). 2

[YJHS12] YÜCER K., JACOBSON A., HORNUNG A., SORKINE O.: Transfusive image manipulation. *ACM Trans. Graph. 31*, 6 (2012). 2

[ZTD*12] ZHAO Q., TAN P., DAI Q., SHEN L., WU E., LIN S.: A closed-form solution to retinex with nonlocal texture constraints. *IEEE Trans. Pattern Anal. Mach. Intell. 34*, 7 (2012), 1437–1444. 8

[ZWS*16] ZHANG F.-L., WANG J., SHECHTMAN E., ZHOU Z.-Y., SHI J.-X., HU S.-M.: Plenopatch: Patch-based plenoptic image manipulation. *IEEE Trans. on Vis. and Comp. Graph.* (2016). 2

Appendix: Fourier Analysis

We extend the Fourier analysis of Bonneel et al. [BTS*15] to our scenario. Similarly, we study the case where w_s and w_t are constant. For the anchor view, the result is the same since we use their technique. Using our notation:

$$\mathcal{F}(O_{at}) = \frac{4\pi^2\xi^2\mathcal{F}(P_{at}) + w_t\mathcal{F}(\mathcal{T}_{at}(O_{a(t-1)}))}{4\pi^2\xi^2 + w_t} \tag{4}$$

where $\mathcal{F}$ denotes the Fourier transform in the image plane and ξ the associated spatial frequency. This shows that low frequencies are dominated by the previous output frame $O_{a(t-1)}$ and the high frequencies by the current processed frame P_{at}.

We now derive a similar result for the secondary views. First, we apply the Euler-Lagrange formula to Equation 2:

$$\begin{aligned} -\Delta O_{it} + (w_t + w_s)O_{it} \\ = -\Delta P_{it} + w_t\mathcal{T}_{it}(O_{i(t-1)}) + w_s\mathcal{S}_{it}(O_{at}) \end{aligned} \tag{5}$$

Then, we apply the Fourier transform and rearrang the terms:

$$\mathcal{F}(O_{it}) = \frac{4\pi^2\xi^2\mathcal{F}(P_{it}) + w_t\mathcal{F}(\mathcal{T}_{it}(O_{i(t-1)})) + w_s\mathcal{F}(\mathcal{S}_{it}(O_{at}))}{4\pi^2\xi^2 + w_t + w_s} \tag{6}$$

The high frequencies are still dominated by the current processed frame P_{it} but the low frequencies come mostly from the previous output frame $O_{i(t-1)}$ and from the current output anchor frame O_{at}, the balance between both being controlled by the weights w_t and w_s.