Eurographics Symposium on Rendering (2007)
Jan Kautz and Sumanta Pattanaik (Editors)

Using Photographs to Enhance Videos of a Static Scene

Pravin Bhat[1] C. Lawrence Zitnick[2] Noah Snavely[1] Aseem Agarwala[3]
Maneesh Agrawala[4] Michael Cohen[1,2] Brian Curless[1] Sing Bing Kang[2]

[1]University of Washington [2]Microsoft Research [3]Adobe Systems [4]University of California, Berkeley

Abstract

We present a framework for automatically enhancing videos of a static scene using a few photographs of the same scene. For example, our system can transfer photographic qualities such as high resolution, high dynamic range and better lighting from the photographs to the video. Additionally, the user can quickly modify the video by editing only a few still images of the scene. Finally, our system allows a user to remove unwanted objects and camera shake from the video. These capabilities are enabled by two technical contributions presented in this paper. First, we make several improvements to a state-of-the-art multiview stereo algorithm in order to compute view-dependent depths using video, photographs, and structure-from-motion data. Second, we present a novel image-based rendering algorithm that can re-render the input video using the appearance of the photographs while preserving certain temporal dynamics such as specularities and dynamic scene lighting.

Categories and Subject Descriptors (according to ACM CCS): I.4.3 [Image Processing and Computer Vision]: Enhancement – Registration I.4.8 [Image Processing and Computer Vision]: Scene Analysis – Stereo, Time-varying imagery

1. Introduction

We have recently witnessed a revolution in the resolution and quality of digital still cameras, with reasonably priced digital cameras now capturing surprisingly high quality photographs. Image editing tools have also become relatively mature and easy to use. In contrast, similarly priced video cameras capture video that is noisier, of lower resolution, and more difficult to edit[†] than digital photographs.

There are many practical reasons for this disparity in progress. Video produces much more data than still photography and thus necessitates more aggressive compression and limited resolution. Still cameras can use flash or long exposure, while video cameras have to generally make do with the available light in the scene and use short exposure times to maintain frame rates. Manual photo editing is commonplace because it is not too time-consuming, whereas the time required to convincingly edit every frame of a video is prohibitive.

Our goal is to bring some of the benefits of still photography to video. We envision the user complementing a video shoot with a few high quality photographs of the scene. In fact, future hardware could be designed to simultaneously capture such datasets. With the right video authoring tool, the user could then transfer the desirable qualities from the photographs to the video. Such an authoring tool could be created given an algorithm that can establish pixel-to-pixel correspondence between the video and the photographs. However, solving this correspondence problem in the general case is incredibly challenging. Scene motion, changes in scene lighting, and differences in viewpoint and camera characteristics can drastically alter the appearance of the scene between the video and photographs.

In this paper, we demonstrate a step towards achieving the broad goal of enhancing real world videos using photographs. To make the problem feasible, we restrict ourselves to scenes with static geometry (i.e., temporal changes in the video are due to camera motion, view-dependent effects like specular reflection, and/or changes in scene lighting). Assuming static geometry allows our system to establish correspondence using structure from motion and multi-view stereo. We note, however, that establishing dense correspondence in the presence of scene motion remains an active area of research in computer vision, and we believe our rendering framework could readily accommodate videos of dynamic scenes given a robust correspondence algorithm.

Leveraging the correspondences computed using the static ge-

[†] In this paper we use *video editing* to refer to the manipulation of pixels in a video rather than the re-ordering of video frames.

(a) Super-resolution (b) High dynamic range (c) Enhanced exposure (d) Object touchup (e) Object removal

Figure 1: *Video enhancements produced by our system. Given a low quality video of a static scene (top row) and a few high quality photographs of the scene, our system can automatically produce a variety of video enhancements (bottom row). Enhancements include the transfer of photographic qualities such as (a) high resolution, (b) high dynamic range, and (c) better exposure from photographs to video. The video can also be edited in a variety of ways by simply editing a few photographs or video frames (d, e).*

ometry assumption, our system provides a general framework for enhancing videos in a number of ways.

Transferring photographic qualities: A user can transfer high resolution, better lighting, and high dynamic range from a few high-quality photographs to the low-quality video. We demonstrate that artifacts commonly found in consumer-level video such as noise, poor exposure, and poor tone-mapping are significantly reduced using our system.

Video editing: A user can edit a video by editing just a few photographs (or, equivalently, a few keyframes of the video). These manual image edits, such as touch-ups, image filters, and objects mattes, are automatically propagated to the entire video.

Object and camera-shake removal: A user can remove unwanted objects from a video by roughly specifying its outline in a few photographs or video frames. We also demonstrate the ability to stabilize video by smoothing out the original camera path and re-rendering the scene as seen from the new camera path.

Our research has *three* main contributions.

First, we show that by augmenting consumer-level video with a few high-quality photographs a unified framework can be created to solve a wide variety of problems that have been previously investigated in isolation.

Next, we present several improvements to Zitnick et al.'s multi-view-stereo (MVS) algorithm [ZK06]. Using numerous real world scenes, we demonstrate that our improved MVS algorithm is able to extract view-dependent depths of a scene containing complex geometry, and do so robustly in the presence of dynamic lighting, noise, and unconstrained camera paths.

Finally, our main technical contribution is a novel image-based rendering (IBR) algorithm that can re-render the input video using the appearance of the photographs. The depths estimated by stereo algorithms are generally imperfect; thus the typical IBR approach of rendering and blending textured depth maps results in blurring or ghosting artifacts. Our IBR algorithm is able to preserve the high-frequency details of the photographs used to reconstruct the video by instead *stitching together* large patches. In addition, using a gradient-domain approach, our IBR algorithm enforces temporal variations in the radiance of a scene-point observed in the input video to be replicated in the enhanced video. As a result, our enhanced videos combine the spatial richness of the input photographs (e.g., high dynamic range, high resolution) with the temporal richness of the input video like moving specularities and dynamic lighting without explicitly modeling these phenomenon.

2. Related Work

The core of our work is a general technique for automatically combining photographs and videos of a static scene to create many interesting effects. Many effects shown in this paper are commonly created by folks in the visual effects industry using software packages such as Shake, BouJou, and AfterEffects. While the degree of automation in these packages has grown considerably in the past decade, they still rely heavily on manual effort (e.g., matting, rotoscoping, etc.) and specialized capture equipment (e.g., blue screen filming, steady cams, use of robotic arms to capture precise video paths, etc.); all are highly unappealing solutions to the amateur user. For example, let us consider the task of editing the surface texture of an object in a video. One could use rotoscoping to track the surface (which could require considerable manual effort) and propagate user edits from a reference video frame to all other video frames. To make matters worse, the rotoscoping option could only be used for near-planar surfaces. This is because a non-planar surface could exhibit self-occlusions and other complex distortions with changes in viewpoints. Alternatively, the user could use the commonly used matchmove feature which allows the insertion of virtual objects into a video while automatically accounting for camera motion. However, this feature would still require the user to model,

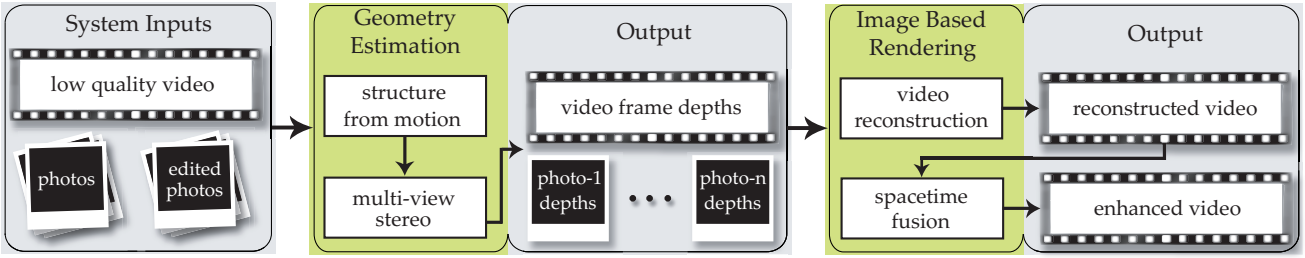


Figure 2: *Our system for enhancing videos using photographs consists of two main components. The geometry estimation component computes view-dependent depths for each video frame and photograph using structure from motion and multi-view stereo. The image-based rendering component uses the estimated geometry to assign each pixel in the video to a corresponding pixel in the photographs. This pixel-to-pixel mapping is used to reconstruct the input video using the original or manually-edited photographs. Finally, our spacetime fusion algorithm improves the result by combining the spatial richness of the reconstructed video with the temporal dynamics of the input video. Figures 3, 6, and 7 show an example sequence with intermediate results produced by individual system components.*

skin, and light the virtual object, and then create an occlusion matte for the final compositing. In contrast, our technique completely automates the creation of many commonly used effects. Admittedly, however, our current technique is restricted to videos of a static scene.

In the research community, day-time photographs have been used to improve the legibility of night-time videos [RIY04], and Shechtman et al. [SCI05] demonstrated video super-resolution using stills. Both of these systems, however, assume no parallax between the video and photographs. There is also a long history of work that takes advantage of multiple videos. Perhaps the closest related work in spirit is Video Epitomes [CFJ05], which can reconstruct portions of a video by first learning its epitome. This representation can be used to reduce noise, fill holes in video, or perform super-resolution (assuming some portions of the video are more zoomed-in than others). Similarly Sawhney et al. [SGH*01] transfer the appearance of a high-resolution video to a nearby low-resolution video. In contrast to these two approaches, we transfer appearance from nearby photographs rather than video, which introduces additional challenges such as temporal consistency.

Another possibility is to use additional images or videos that are not necessarily of the same scene as examples for a machine learning approach. This approach has been demonstrated for super-resolution of images [FJP02] and video [BBM03]. In contrast, our task is easier because we are given high-resolution examples of the *same* scene. Not surprisingly, our results are much more faithful to the real scene. The analogies approach learns a transformation between two images [HJO*01] or two videos [HE02]; our *video editing* application is similar to these methods as it uses analogy-style user input, though unlike previous work we apply transformations defined on images to videos.

Several recent techniques are designed to fuse multiple photographs or videos to improve the resulting depictions. Examples for photographs include interactive photomontage [ADA*04], combined long and short exposures [JSTS04], and combined flash/no-flash [PAH*04]. Sand and Teller [ST04] describe a registration algorithm for fusing two videos shot along similar camera paths. In constrast we fuse photographs and video to produce video, and do not restrict camera paths.

The two core algorithms in our system, multi-view stereo (MVS) and image based rendering (IBR), have many antecedents. Stereo continues to be an active area of research for both two-views [SS02] and multiple views [SCD*06]; our MVS algorithm is an extension of the algorithm proposed by Zitnick et al. [ZK06]. Image-based rendering and novel-view interpolation [KS02] continue to receive considerable attention; we describe a novel IBR approach that uses both graph-cuts [BVZ01] and gradient-domain compositing [PGB03] while taking camera motion into account. Our graphcuts compositing preserves high-frequency details of the images used for reconstruction (e.g., photographs) while the our gradient-domain compositing hides spatial and temporal seams between the images. Moreover, if the camera path of the output video matches the camera path of the input video then our IBR algorithm can also incorporate the view-and-time dependent lighting effects captured by the input video.

Many of the results we demonstrate can be generated using other specialized techniques. Examples include producing high-dynamic range video [KUWS03], correcting poorly-exposed video [BM05], creating object mattes [WBC*05], consistently editing an object in multiple images [SK98], removing objects by filling holes in video [WSI04], and IBR-based video stabilization [BBM01]. Our approach makes the trade-off of restricting to static scenes and optionally including photographs to give a single (and, we believe, extensible) framework that achieves each of these effects.

3. Overview

Figure 2 provides a high level overview of our system pipeline. The inputs to the system depend on how the user wishes to enhance the video. To transfer photographic qualities, the user inputs both a video and a few photographs of the scene (see Figure 3). For other enhancements the user can choose to input pho-

Input photographs Input and output videos

Figure 3: *Example inputs and output of our system. Given a video with overexposed regions (top row) and a few photographs of the scene shot with proper exposure (left column), our system automatically produces an enhanced video (bottom row). Only two of the eleven input photographs have been shown here.*

tographs, or to simply choose a few keyframes of the video to serve the same purpose. For example, to perform object removal, the user specifies a rough outline of the object in a few video frames or photographs. Similarly, to perform video editing the user provides registered pairs of original and edited images (similar to the user input used by Image Analogies [HJO*01]); the image-pairs may be created using photographs or video frames.

After the inputs are specified, the system performs geometry estimation and image based rendering to create an enhanced video. In the remainder of this paper we focus on the details of these two system components, and then present a variety of video enhancements produced by our system. Finally, we conclude by discussing the limitations of our approach and suggesting areas of future research.

4. Geometry Estimation

The first phase of our pipeline consists of estimating the geometry of the scene imaged by the video and photographs. We begin by recovering camera poses and sparse scene geometry using a structure from motion (SFM) algorithm. Then, we produce a view-dependent depth map (example shown in Figure 6) for each photograph and video frame using a multi-view stereo (MVS) algorithm.

4.1. Structure from motion

Our system uses the structure-from-motion implementation of Snavely et al. [SSS06]. Like any standard structure-from-motion library, Snavely's system produces projection matrices for each photograph and video frame, a sparse cloud of 3D scene points, and a list of the viewpoints from which each scene point is visible.

4.2. Multi-View Stereo

Our multi-view stereo algorithm extends the approach proposed by Zitnick et al. [ZK06]. We choose to build on their algorithm because it is specifically tailored for image based-rendering of scenes constructed from video footage. They construct view dependent depths maps using an over-segmentation approach that is robust to image noise and some types of scene specularities. We make three improvements on their work: (1) we specialize our algorithm for heterogeneous datasets containing both video frames and photographs; (2) we use a wider range of disparity planes resulting in higher quality depth maps; and (3) we use 3D scene points generated by the structure from motion system to improve depth estimation.

Before elaborating on the various improvements we are proposing, here is a brief introduction to the previous algorithm. In the first step, each input image is divided into segments using a color-based over-segmentation algorithm. The second step computes a disparity for each segment. This step is performed by constructing a pair-wise Markov Random Field (MRF) for each image. Each node in the MRF corresponds to a segment, and edges are added between all nodes corresponding to abutting segments. Similar to the color consistency constraint used in pixel-based MRFs, the prior of the MRF biases neighboring segments with similar average colors to have similar depths. The likelihood or data term of the MRF is computed using a standard color similarity measure, as well as incorporating occlusion information based on current disparity beliefs. The disparity beliefs of all segments for all images are simultaneously updated using loopy belief propagation. At each iteration, the data term of each MRF is biased to be consistent with the beliefs of MRFs from neighboring images. Neighboring MRFs are also used to compute occlusion information. As a result, information is shared between all neighboring images, and the final disparities are largely consistent.

It should be noted that we do not expect the reader to be able to implement our MVS algorithm without reading Zitnick et al.'s paper [ZK06] that describes the previous algorithm in detail. Though a full understanding of the previous algorithm is

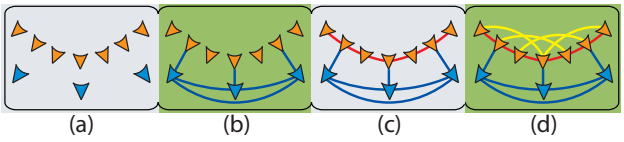


Figure 4: *An overview of viewpoint neighbors: (a) Viewpoints corresponding to video frames (orange nodes) and photographs (blue nodes). (b) Each photograph is connected as a neighbor to its two nearest photographs and its nearest video frame (blue links). (c) Each video frame is connected to its two nearest video frames (red links) and (d) two video frames that are M frames away (yellow links). Here M is three, though we typically set M to ten. Distance between viewpoints is determined using our viewpoint distance metric.*

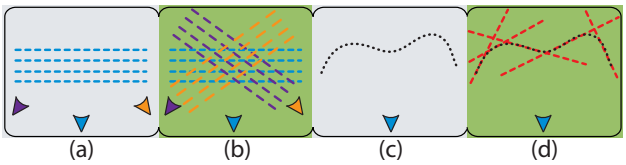


Figure 5: *Options for computing disparity planes: (a) Disparity planes generated using planes parallel to a reference viewpoint (blue camera). (b) Disparity planes generated using planes parallel to each viewpoint (i.e., view-dependent disparity planes). (c) 3D scene points generated by SFM. (d) Non-fronto-parallel disparity planes generated to approximate the 3D scene points.*

not required to appreciate the various improvements we will now present.

4.2.1. Handling video and photographs

Working with both photos and videos requires special attention to color matching, construction of the viewpoint neighborhood graph, and the definition of a metric for proximity of viewpoints that incorporates similarity of fields of view.

4.2.1.1. Determining viewpoint neighbors The previous algorithm assumes that neighboring viewpoints are defined by the user based on the camera setup. We automate this step. For each viewpoint, the neighboring viewpoints are determined based on 2 objectives. First, the neighborhood graph (created by treating viewpoints as nodes and pairs of neighboring viewpoints as edges), should avoid disconnected components. This objective ensures that depth information can propagate between all viewpoints through the data term of the MRF. Second, viewpoints should be considered neighbhors when they facilitate robust stereo matching. If only very close viewpoints are matched as neighbors, the small baselines may lead to ambiguous and low-precision depths. Alternatively, only matching viewpoints that are far from one another will lead to problems due to occlusions. We thus pair each viewpoint with a mixture of nearby and distant viewpoints, as illustrated in Figure 4 and described in its caption.

4.2.1.2. Viewpoint distance metric One could use the Euclidean distance between viewpoints to evaluate which views are near or far from each other. However, this metric can perform poorly when, for example, two nearby cameras point in opposite directions. We instead measure distance as the inverse of the number of 3D scene points observed in common by two viewpoints. In the absence of scene point information the number of feature matches between two images can also be used to compute distance. Our distance metric can be seen as measuring the amount of overlap in scene structure observed by two viewpoints.

4.2.1.3. Color matching The video frames and photographs generally have different color characteristics due to differences in lighting and reflection, exposure, and sensor characteristics, making it more difficult to compute correspondences. Thus, when computing the data term for any reference view, we first match the color distribution of each neighboring view to that reference view using the color-matching algorithm proposed by Reinhard et al. [RAGS01].

4.2.2. Augmenting the set of disparity planes

To improve the ability of the multi-view stereo algorithm to reconstruct the depths of observed surfaces, we augment the set of available disparity planes in two ways.

4.2.2.1. View-dependent disparity planes In the previous algorithm, the set of possible segment depths for each viewpoint are derived from a set of disparity planes that are parallel to the projection plane of a single reference viewpoint (Figure 5a). The more orthogonal a camera's viewing axis is to the reference camera's viewing axis, the more difficult it is for this set of disparity planes to adequately approximate the depths for that camera. In our algorithm, each viewpoint uses a set of unique disparity planes that are parallel to its own projection plane (Figure 5b). In addition to improving the depth estimation, this modification allows our algorithm to handle camera paths that could not be handled by the previous algorithm, such as a 360° loop around an object. It should be noted that view-dependent disparity planes are *not* a novel contribution on our part and have been used in other stereo algorithms [KS04].

4.2.2.2. Non-fronto-parallel disparity planes Many surfaces in real-world scenes are slanted planes, e.g., floors and walls. Though fronto-parallel disparity planes allow reconstruction of arbitrary shape, these slanted planes can be more succinctly described using non-fronto-parallel disparity planes. Thus, we augment the initial set of disparity planes with several slanted planes that are recovered from the 3D scene points (Figure 5c, 5d) using iterative RANSAC [Tor98]. There has been some prior work in using non-fronto-parallel disparity planes in MVS algorithms [YWY*06, KSK06]. These methods extract non-fronto-parallel planes by matching small texture patches between two neighboring viewpoints. In contrast, using the 3D scene points

generated by the SFM algorithm allows our plane extraction method to incorporate information from all viewpoints.

4.2.3. Incorporating SFM scene points

Structure from motion yields a sparse estimate of 3D scene structure that can be used as a prior in the multi-view stereo algorithm. Each 3D scene point is associated with a list of viewpoints that observe it. If we project a scene point onto a segment in each of these viewpoints, the depth of the segment should be consistent with the depth of the scene point. Thus, we modify the data term in the MRF to minimize the distance between each 3D scene point and the hypothesized disparity plane of the segment onto which the 3D scene point projects.

5. Image Based Rendering

The view-dependent depths computed in the first phase of our system implicitly define a correspondence between each pixel in the video and pixels in one or more photographs, as well as a correspondence between pixels in consecutive frames of video. In the second phase of our system, we use this correspondence to reconstruct the video using the appearance of the photographs while preserving the temporal dynamics of the video. The first algorithm in this phase, which we call *video reconstruction*, uses an MRF formulation to choose a photograph from which to copy color for each video pixel, with the goal of producing a plausible and seamless composite. The second algorithm, which we call *spacetime fusion*, produces the final result by integrating a gradient field created using the spatial gradients of the video-reconstruction result and temporal gradients of the input video. The gradient field is defined in a manner that enforces the temporal variations in the radiance of a scene-point observed in the input video to be replicated in the final result. As a consequence, the final result mimics the temporal variations of the input video (e.g., dynamic lighting, moving specularities, etc) and avoids temporal incoherence caused by errors in the depth estimates and large exposure changes in the source photographs.

5.1. Video Reconstruction

Since the projection matrix of each video frame is known, one approach to reconstructing the video would be to perform novel view interpolation (e.g., [ZKU*04, HKP*99]) from the photographs for each video frame. However, typical view interpolation algorithms produce a result that is a weighted average of the warped input images, which generally (and in our experience) result in ghosting artifacts and loss of high-frequency details. Instead, we use an MRF formulation to create a composite from large, coherent patches of the projected inputs while minimizing the visibility of the patch boundaries. Also, unlike the classic view interpolation problem, we have depth and color information (albeit of lower quality) for each video frame we are trying to interpolate (reconstruct) from the input photographs. This information can be used to guide the choice of which photograph to use for a particular video pixel. In particular, we can favor source photograph pixels that have similar depth and color to those of the existing video pixel to resolve occlusions and promote visually faithful reconstructions.

We will now describe the version of the video reconstruction algorithm designed for enhancements that transfer photographic qualities to video. The minor modifications required for other video enhancements are described later in this section.

Given a video frame V_i, we begin by selecting a set of source views $S_{1..N}$ to be used in its reconstruction. $S_{1..N}$ is formed of the N nearest photographs to V_i. We use $N = 4$, except when doing object and camera-shake removal, as described later in this section. We determine the nearest photographs using the viewpoint distance metric defined in Section 4.2.1. Using a z-buffered point splatting algorithm, we then separately re-render each source view S_j from the viewpoint of video frame V_i. The result is a set of N re-projected images $P_{1..N}$ and corresponding re-projected depth maps. Each image P_i provides each pixel in V_i with one or zero reconstruction candidates (P_i may be undefined in some regions of V_i). Note that, for the case of superresolution, we spatially upsample the video frames and their depth maps as a pre-process.

The video reconstruction problem can now be formulated as a labeling problem in an MRF framework. In particular, the goal is to assign to each pixel $p \in V_i$ a label $L(p)$ indicating which of the N reconstruction candidates provided by $P_{1..N}$ should contribute to the final result. Ideally, we would optimize the labeling of all video frames in a single MRF formulation instead of constructing an MRF for each frame independently. However, inference on a 3D MRF can be computationally expensive, and we have found experimentally that acceptable results are produced by computing each frame independently. The per-frame approach is able to maintain some temporal coherence because the multi-view stereo algorithm encourages depths in each video frame to be consistent with the depths in neighboring video frames. Any residual temporal incoherence in the reconstructed video is removed by the spacetime fusion algorithm.

Our cost function to evaluate the quality of a given labeling L is in the form of a standard Markov Random Field:

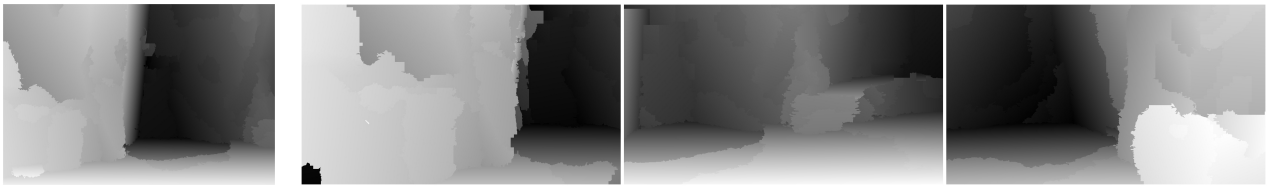
$$C(L) = \sum_{p \in V_i} C_{\mathrm{D}}(p, L(p)) + \lambda \sum_{\{p,q\} \in \mathcal{N}} C_{\mathrm{S}}(p, q, L(p), L(q)), \quad (1)$$

where C_{D} is the data cost function, C_{S} is the smoothness cost function, p and q are neighboring pixels defined by $\mathcal{N}$ which is the set of all eight-connected neighbors in V_i, and λ is a user defined parameter that controls the trade-off between data cost and smoothness cost. We typically set λ to 2.0.

The data cost function encourages video pixels to be reconstructed from photographs with similar color and depth. We define C_{D} as:

$$C_{\mathrm{D}}(p, L(p)) = \begin{cases} \infty & \text{if } P_{L(p)}(p) \text{ is undefined} \\ \infty & \text{if } |d(V_i, p) - d(P_{L(p)}, p)| > \kappa \\ ||V_i(p) - P_{L(p)}(p)|| & \end{cases},$$

where $P_{L(p)}$ is the projection associated with $L(p)$, $d(I, p)$ denotes the depth value at pixel p in I, and κ is a user defined constant that controls the mismatch between the depth of a pixel and its depth



Photograph depths Video depths

Figure 6: *View-dependent depth maps generated by our multi-view stereo algorithm for the images in the top row of Figure 3.*

in a projected image. We typically set κ to a value that corresponds to a projection error of 5 pixels.

To make the function C_D more robust to differences in the color distribution of V_i and $P_{1..N}$, we first match the color distribution of each P_i to the color distribution of V_i using the method of Reinhard et al. [RAGS01]. This step is repeated for each frame i. Note that these transformed colors are only used for computing C_D and are not used in creating the final composite.

In Equation (1), C_S measures how well the labeling produces a seamless reconstruction. C_S is similar to the smoothness cost introduced by Kwatra et al. [KSE*03], and is defined using two terms, X and Y:

$$X = ||P_{L(p)}(p) - P_{L(q)}(p)|| + ||P_{L(p)}(q) - P_{L(q)}(q)||,$$

$$Y = 2.0 - (\nabla_{p,q} P_{L(p)} + \nabla_{p,q} P_{L(q)}),$$

where image color channel values are in the range $[0..1]$ and $\nabla_{p,q} P_i$ returns the gradient magnitude between pixels p and q in P_i. Lastly, $C_S(p,q,L(p),L(q)) = X * Y$. Thus, C_S encourages a seam between two projections to either pass through regions where the projections have similar colors (i.e., term X), or to run along strong edges (i.e., term Y).

We assign each pixel $p \in V_i$ to one of its candidate labels by using alpha-expansion moves [BVZ01] to find a labeling that is the approximate global minimum of our cost function. Note that for some pixels in V_i the cost is infinite, which means that no reprojected depths were a suitable match; this can occur if the corresponding surface point was not observed in any of the source photographs. Such pixels are not assigned a label, resulting in holes in the reconstructed video that are later filled in by the spacetime fusion algorithm.

5.1.1. Video reconstruction specializations

The above algorithm can be applied to transfer qualities from photographs to video. For other applications of our framework, however, some modifications are required.

Object Removal: To remove an object from a video, the user specifies a rough mask for the object in one or more source images, which may be photographs or video frames. The masks are drawn to inscribe the object (i.e., no pixels outside the object boundary are selected). The union of the masks is then transferred to every video frame and dilated in order to select the entire object (and possibly a few background pixels near the object boundary). Now the goal is to reconstruct the masked portion of each video frame using the unmasked portion of nearby video frames and photographs.

Remember, the video reconstruction algorithm assumes that the approximate depths and colors of the region being reconstructed is avaliable before the reconstruction. However, the color and depth information for our target regions (i.e., masked portion of each video frame) is not avaiable. We approximate this color and depth information by re-projecting the unmasked portions of a large number of nearby video frames and photographs (typically 75) into each video frame's masked area, resulting in several candidates for each masked video pixel. We set the depth and color of each masked pixel to the median value of its candidates. Note that these values do not need to be very precise, since they are only used as a guide to the video reconstruction algorithm when selecting among source images for a video pixel. Now we can use the video reconstruction algorithm to reconstruct the masked region of the video frame. The source views $S_{1..N}$ for each video frame in the video reconstruction algorithm are generated using the unmasked portions of the 75 nearest video frames and photographs.

Camera-shake removal: To remove camera shake from an input video our system creates a new camera path by smoothing the recovered extrinsic camera parameters (translation and rotation) of nearby video frames. Unfortunately, the result of this operation is that color and depth information is not available for the new viewpoints we wish to reconstruct, though color and depth are available for nearby views (i.e., original video frames). We use the approach described in the object removal case to approximate the color and depth information in the new viewpoints, and then apply the video reconstruction algorithm to reconstruct each new viewpoint using 20 nearest video frames and photographs as source images. Since the new viewpoints have not been captured in any of the input images we cannot use our viewpoint distance metric to determine the distance between two viewpoints. Instead we simply use the Euclidean distance between the viewpoints as determined by their extrinsic camera parameters.

Video editing: When performing video edits, each input photograph S_j is accompanied by a user-edited version of the photograph, S'_j. We would like the video reconstruction algorithm to produce a video that is constructed using the pixel in $S'_{1..N}$. So we project these images to form reconstruction candidates $P'_{1..N}$

for each video frame V_i, just as we projected $S_{1..N}$ to form $P_{1..N}$. To avoid visible seams in the output, the smoothness cost C_S is computed from the colors of $P'_{1..N}$. The data cost C_D, however, is computed from the unedited colors of $P_{1..N}$ since the data cost measures how well the photographs and unedited video frames match.

A problem with this approach is that it is unreasonable to expect the user to manually edit every photograph in the input and, moreover, to edit each photograph in a perfectly consistent fashion; that is, some of the images may not be edited and, in images that are edited, the edited pixels in different images that correspond to the same real-world location may not have the same color. These differences can lead to flickering in the final video. To mitigate this problem we propagate user edits across the photographs to ensure that they are consistent before running the video reconstruction algorithm. Starting with one of the edited photographs S'_j, call it the "sender," we re-project its depths into its neighboring photographs, call them "receivers." At each pixel in receiver S_k, if the re-projected depth is within κ of the receiver depth, then we copy the re-projected color of S'_j into S'_k. The receivers are then added to a FIFO queue, and the process is repeated, drawing from the queue until it is empty. Once a photograph has been a sender, it is never reused as a sender or receiver. Figure 7 shows a video editing result where the user has applied a painterly effect that randomly places brush strokes onto each photograph. In this case the painterly effect across the photographs is made consistent using our edit propagation mechanism. Then the video reconstruction and spacetime fusion algorithms are used to transfer the painterly effect from the photographs to video. As a result brush strokes appear to adhere to the surfaces in the video without temporal inconsistency (see supplementary video).

5.2. Spacetime Fusion

The video created by the video reconstruction algorithm will resemble the photographs, but may suffer from several artifacts. The main problem of concern is that a video reconstructed using photographs often appears oddly lifeless, because it lacks the rich temporal variations found in videos even of static scenes; for example, the position of specular highlights will not move as the viewpoint changes. In addition, spatial and temporal seams between various photographs used to reconstruct the video may still be visible due to large exposure variations in the photographs. Also, holes may appear in the areas of the reconstructed video where none of the projections were defined (e.g., portions of the scene not seen in any of the photographs).

A common approach to compositing regions from several photographs (or videos) while minimizing visible seams is to combine them in the gradient domain [PGB03, WRA04]. Typically, gradient-domain compositing is performed by copying gradients rather than colors from the source imagery to form a gradient field $G(x,y,t)$. Then, the enhanced video $E(x,y,t)$ is created by solving for a video whose gradient field is as close as possible to $G(x,y,t)$. Our spacetime fusion algorithm is similar in spirit to this approach.

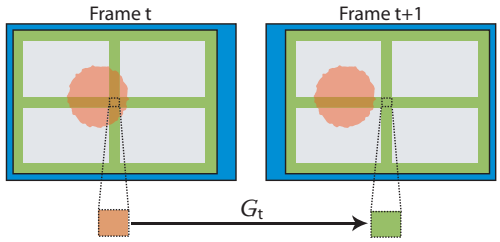


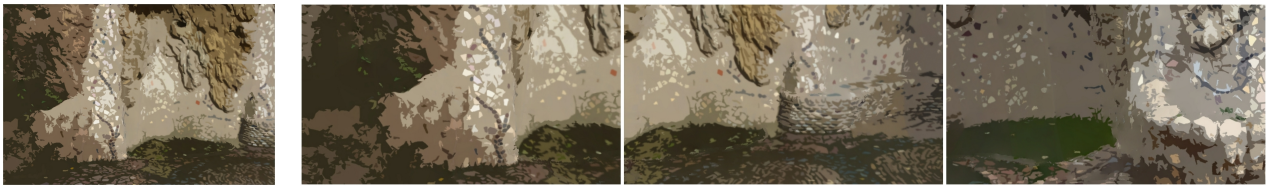
Figure 8: *The figure depicts a glass window (with shiny frame) seen in two consecutive video frames. A change in camera viewpoint causes the reflection of the sun (shown in red) to shift with respect to the window pane. Using pixel depths we can determine all pairs of pixels in the two frames that correspond to the same real world location (e.g., the pixel-pair outlined with a dotted square). The spacetime fusion algorithm constrains the temporal gradient between such pixel-pairs to match the input video, thus preserving the dynamics of the reflection.*

For our applications, we wish the output to exhibit the rich spatial properties of the photographs, as well as, the temporal variations of the input video. We modify the standard gradient-domain composing methods, by using the motion-compensated temporal gradients G_t from the input video, and spatial gradients G_x and G_y from the photographs.

Specifically, we define the spatial gradients as follows. In areas where the labeling of a video frame is undefined (i.e., holes) we copy spatial gradients from the input video. To improve the color consistency, we first transform colors in the hole region by color matching the original frame to the non-hole portions of the reconstructed frame (again using the color distribution matching method of Reinhard et al. [RAGS01]). In areas where the labeling of a video-frame transitions from one source to another (i.e., a photograph-to-photograph or photograph-to-hole transition), the pixels used in computing the gradient may come from different sources; in this case, we average the spatial gradients of the two overlapping sources at the transition seam. Spatial gradients in all other regions are copied directly from the reconstructed video.

The temporal gradients, on the other hand, are created from the original input video *after* accounting for motion. We have found that using Wang et al.'s [WRA04] approach of constraining temporal gradients between temporally adjacent pixels (i.e., between pixels (x,y,t) and $(x,y,t+1)$) leads to severe ghosting artifacts for videos with camera motion. Also, to properly capture the temporal variations in the input video, we need to constrain the temporal gradient between input video pixels that correspond to the same real-world location (Figure 8). Therefore, the temporal gradient $G_t(x,y,t)$ is defined using the pixel at (x,y,t) in the input video and its corresponding pixel in frame $t+1$. The correspondence is computed by re-projecting the pixel (with depth) at time t into the viewpoint of the frame at time $t+1$.

We create the enhanced video E by solving an over-constrained system of linear equations formed by the following three con-



Edited photograph Edited video

Figure 7: *A video editing example: (Left column) one of the eleven input photographs processed using a painterly effect found in Adobe Photoshop; (Right column) painterly video created by transfering the painterly effect from the photographs to the input video.*

straints for each pixel in V:

$$E(x+1,y,t) - E(x,y,t) = G_x(x,y,t),$$

$$E(x,y+1,t) - E(x,y,t) = G_y(x,y,t),$$

$$E(x+u,y+v,t+1) - E(x,y,t) = G_t(x,y,t),$$

where (u,v) is a correspondence vector linking the pixel at (x,y,t) and its corresponding pixel in frame $t+1$. (Each color channel forms an independent set of linear equations.)

Since (u,v) is a non-discrete motion vector and the variables in the linear system correspond to discrete locations, (u,v) must somehow be discretized. One option is to simply round (u,v) to the nearest integers. We have obtained more accurate results, however, using a form of bi-linear interpolation. That is, each correspondence vector results in temporal constraints for the four integer coordinates nearest to $(x+u,y+v,t+1)$, weighted by the distance between the floating-point location and the integer location. We further weight each temporal constraint by $A(x,y,t) * \tau$, where $A(x,y,t)$ is a measure of the confidence in the accuracy of (u,v), and τ is a user-defined constant that controls the trade-off between fidelity to the spatial gradients and the temporal gradients. $A(x,y,t)$ is set to the probability of the depth assigned to (x,y,t) during multi-view stereo phase of geometry estimation. The results in this paper were generated using a value between 7.0 and 9.0 for τ.

We use the LASPack [Ska95] conjugate gradient solver to solve the over-constrained linear system. Constraints involving pixels outside the boundaries of the video spacetime volume are modeled using Neumann boundary conditions. Large videos that cannot be solved in main memory are solved in slabs of 20-30 frames. When solving a slab of frames, the boundary conditions for faces shared with adjacent slabs are modeled using Dirichlet boundary conditions, while the remaining faces continue to use Neumann boundary conditions. Using mixed boundary conditions in this manner allows information to flow through the entire video during optimization and ensures temporal coherency between adjacent slabs in the final result.

5.2.1. Spacetime fusion specializations

In some cases, the user might not want to transfer the temporal variations from the input video to the enhanced video (e.g., if the input video exhibits noise or other unwanted temporal variations), but may still wish to remove other artifacts in the reconstructed video using spacetime fusion; mainly temporal flickering caused by large exposure variations in the photographs. We can remove these artifacts by assuming pixels that correspond to the same real-world location exhibit constant brightness between adjacent video-frames. That is, the motion-compensated temporal gradients $G_t(x,y,t)$ are set to zero for all x, y, and t. The spacetime fusion result in Figure 3 was generated in this manner to avoid transferring the abrupt changes in sunlight from the input video to the enhanced video.

Finally, when performing spacetime fusion for the *object removal* case we cannot use the depth and color information from the masked regions of the input video since these regions contain the object we want to remove. Instead, we define the motion vectors (u,v) for these regions using the depths projected from the neighboring frames that were used to fill in the color information during video reconstruction (Section 5.1.1). Then, we assume scene points in the masked regions exhibit constant brightness, and set the motion-compensated temporal gradients for these regions to zero. The motion vectors and temporal gradients for the *camera shake removal* case are defined in exactly the same manner.

6. Video Enhancements

Once a mechanism is developed that can register photographs to a video and then blend the two data sources in a seamless fashion, the same mechanism can be used in a wide variety of applications. The following are a few of the video-enhancement applications we have explored using with our system.

Super-Resolution. Figure 1a shows an example where we effectively quadruple the resolution of a video in each spatial dimension using seven high-resolution photographs. For efficiency, the photographs were scaled down to the size of the video frames before computing depths. To generate high-resolution video frames, depths from the low-resolution photographs were upsampled to match the output resolution. Color data was obtained from the original high-resolution photographs.

HDR Video. Figure 1b shows a video of a scene with high dynamic range. Due to the enormous contrast in the scene, no single exposure setting can faithfully capture all the subtleties of

(a) Input video (b) Video matte (c) Selective decolorization (d) Object touch-up (e) Object touch-up

Figure 9: *Video editing examples: Using the depths generated for the input video (a), and a user-drawn object-matte in a single video frame, the system can produce an object-matte for the entire video (b). (c) The object-matte can be used for a variety of effects like selective decolorization. (d) Any touch-ups made to an object in one or more video frames can be consistently applied to the entire video; in this case, reflections on top of the photos are removed. (e) Our system can also apply touch-ups to an object while preserving the lighting effects exhibited by the original object; in this case, the photos are changed without modifying the reflections.*

the scene geometry. We augmented our video capture session with six sets of exposure bracketed photographs which were used to create six HDR photographs. Using the HDR photographs our system increased the dynamic range of the input video.

Enhanced exposure. The input video in (Figure 1c) shows a dinner-table scene lit by light cast through a window on a rainy night. The video contains view-dependent and dynamic lighting effects. Rain drops flowing down the window pane cast dynamic caustics on the dinner table. Also, the silverware in the scene exhibits view-dependent specular highlights. Unfortunately, due to the limited light in the scene the video is severely underexposed in some regions. To improve the exposure of the video we captured twelve photographs of the scene using longer exposure time than it is possible when capturing a video. The input video and photographs were processed by our system to produce a video with enhanced exposure that exhibits the same temporal dynamics as the original video. The example in Figure 3 shows another exposure enhancement result where a video with overexposed regions was enhanced using photographs shot with proper exposure.

Video Editing. Figures 1d, 7, and 9 show a variety of video editing operations that can be used to enhance a video.

Matte generation. Figure 9b shows a result where a user-drawn object matte in a single video frame is used to automatically create an object matte for the entire video. The resulting video matte is consistent with the occlusions in the scene. Such a video matte can be used to create a variety of effects like selectively decolorizing the selected object (Figure 9c).

Object touch-up. In the example shown in Figure 1d we used a texture synthesis algorithm to remove the scar from the tree trunk in a single video frame. Then our system used the modified pixels in the video frame to consistently edit the entire video. Similarly, to remove the reflections on the framed photos (Figure 9d) we replaced each photo with a reflection free image in a single video frame. We also produced the converse result where the photos in the video were replaced with different photos while preserving the reflections (Figure 9e).

Painterly video. Several image filters when applied to video frames independently produce a video that is temporally incoherent; most notable of these filters are painterly effects that "paint" a given image using varied brush strokes. To create a painterly version of the video shown in Figure 3 we first applied a painterly filter to all input photographs. The inconsistencies in the filtered photographs were then removed using the edit propagation mechanism described in section 5.2.1. The painterly filter was then transferred from the filtered photographs to the video using our IBR algorithm to produce a temporally-coherent result (see Figure 7 and supplementary video).

Object removal. Figure 1e shows an example where a no-parking sign is occluding much of the scenic background. We specified the object to be removed, in this case the no-parking sign, by drawing a rough mask in a single video frame. The system then re-renders the input video without the no-parking sign by using the object removal technique described in section 5.2.1.

Camera shake removal. Our system is also able to remove camera shake from a jerky input video by first locally smoothing the projection matrices of the input camera path to create a smooth camera path. A stabilized video is then created by rendering the scene as seen from the smooth camera path using our IBR algorithm (see supplementary video). Unlike traditional video-stabilization methods, which use 2D stabilization (or piecewise 2D stabilization), our method has the advantage of handling videos with significant depth changes and camera shake caused by 3D camera rotation. Also, our method incorporates information from several video frames to fill in information at frame borders to avoid cropping–a common problem with 2D techniques. While our method is not the first to use 3D scene reconstruction for video stabilization (see Buehler et al. [BBM01]), our results are likely to be of higher quality due to advances made by our MVS and IBR algorithms.

All enhancement examples were generated using the same parameter values listed in the paper with the exception of the τ parameter in spacetime fusion. For some examples we generated three spacetime-fusion results in parallel using three different settings for τ (τ settings were picked from the range listed in Section 5.2) and used the most visually pleasing result.

7. Discussion and future work

To demonstrate the versatility of our system we chose to qualitatively test it on numerous real-world examples. Our supplementary video demonstrates eleven results generated using eight

different scenes. Five of the eight input videos suffer from problems like low resolution, specularities, improper exposure, and dynamic lighting. Our experiments demonstrate that using photographs, along with low-quality video, can produce better depth estimates than using the video alone since our MVS algorithm propagates depth information across all views. In addition, the estimated depths need not be perfect as our IBR algorithm can substantially reduce reconstruction artifacts caused by erroneous depths (see our supplementary video).

While our results look visually pleasing, under careful scrutiny some artifacts can be seen in most of the results. These artifacts can be traced to errors in the output of the various computer vision algorithms used in our system. For example, structure-from-motion sometimes yields imprecise projection matrices, which leads to incorrect epipolar constraints being used in the depth estimation. Some artifacts are caused when the image over-segmentation algorithm produces segments that straddle depth discontinuities thus violating the MVS assumption that each segment can be assigned to a single disparity plane. Also, the depths estimated by our MVS algorithm are rarely perfect, and these errors can also lead to artifacts. Lastly, the spacetime fusion algorithm can sometimes introduce a hazy blur in the result video when the depths used to compute the temporal gradient are imprecise. In general, our IBR algorithm is robust to modest errors in depth estimation, but large errors can result in artifacts.

7.1. Future Work

7.1.1. Processing speed

The current processing speed of our system is quite slow with five minutes being spent on each video frame (resolution: 853 x 480); where two minutes are spent on SFM, two minutes are spent on MVS and the last minute is spent on IBR. There is a lot of room for improvement in our unoptimized research code since runtime speed was not our primary concern. Recent work on computing SFM for video with real-time performance [Nis05] would be especially beneficial to our system. Our spacetime fusion algorithm can probably be sped up using a preconditioner similar to the one proposed by Szeliski [Sze06].

7.1.2. User interaction

While computer vision algorithms will continue to improve, we cannot expect them to always perform perfectly. In this paper we show how far automatic algorithms can be pushed; however, a production-quality system would need to incorporate user interaction to further improve the results. Ideally, users could locally fix problems and the system would propagate those fixes across the output. We plan to explore user interaction within the context of our system as future research.

7.1.3. Dynamic scenes

We have demonstrated that video can be enhanced in a number of useful ways by capturing and incorporating several photographs of the scene. We have also described a framework that can achieve these improvements. However, the system described in this paper is only the first step in achieving our overall goal, since most videos that people take depict dynamic scenes. It is important to note that our restriction to static geometry is solely due to our method for computing correspondences; our IBR algorithm for video editing and transferring photographic qualities, for example, can be integrated with any method that can compute correspondence, even for dynamic scenes. We see promise in the advances being made in the general correspondence problem, such as long-range optical flow [WB04], non-rigid shape reconstruction [BZS*06] and synchronized camera arrays [WJV*05]. As computer vision algorithms continue to improve, we believe that the static-scene restriction can be lifted, and that our overarching goal of generally using photographs to enhance videos can be realized.

Acknowledgements: Thanks to Dan Goldman and the reviewers for their insightful comments. Thanks to Kevin Chiu, Wilmot Li, and Tina Loucks for their help with the manuscript. Thanks to Prasad Bhat for creating the picture frame design used in the example shown in Figure 9e. This research was partly funded by an NVIDIA fellowship.

References

[ADA*04] AGARWALA A., DONTCHEVA M., AGRAWALA M., DRUCKER S., COLBURN A., CURLESS B., SALESIN D., COHEN M.: Interactive digital photomontage. In *ACM SIGGRAPH* (2004), pp. 294–302.

[BBM01] BUEHLER C., BOSSE M., MCMILLAN L.: Non-metric image-based rendering for video stabilization. In *2001 Conference on Computer Vision and Pattern Recognition (CVPR 2001)* (Dec. 2001), pp. 609–614.

[BBM03] BISHOP C., BLAKE A., MARTHI B.: Super-resolution enhancement of video. In *International Conference on Artificial Intelligence and Statistics* (January 2003).

[BM05] BENNETT E. P., MCMILLAN L.: Video enhancement using per-pixel virtual exposures. *ACM SIGGRAPH* (Aug. 2005), 845–852.

[BVZ01] BOYKOV Y., VEKSLER O., ZABIH R.: Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell. 23*, 11 (2001), 1222–1239.

[BZS*06] BHAT P., ZHENG K. C., SNAVELY N., AGARWALA A., AGRAWALA M., COHEN M. F., CURLESS B.: Piecewise image registration in the presence of multiple large motions. In *CVPR '06* (2006), IEEE Computer Society, pp. 2491–2497.

[CFJ05] CHEUNG V., FREY B., JOJIC N.: Video epitomes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2005), vol. 1, pp. 42–49.

[FJP02] FREEMAN W. T., JONES T. R., PASZTOR E. C.: Example-based super-resolution. *IEEE Computer Graphics & Applications 22*, 2 (March-April 2002), 56–65.

[HE02] HARO A., ESSA I.: Learning video processing by example. In *International Conference on Pattern Recognition* (2002).

[HJO*01] HERTZMANN A., JACOBS C., OLIVER N., CURLESS B., SALESIN D.: Image analogies. In *ACM SIGGRAPH* (2001).

[HKP*99] HEIGL B., KOCH R., POLLEFEYS M., DENZLER J., GOOL L. J. V.: Plenoptic modeling and rendering from image sequences taken by hand-held camera. In *DAGM-Symposium* (1999), pp. 94–101.

[JSTS04] JIA J., SUN J., TANG C.-K., SHUM H.-Y.: Bayesian correction of image intensity with spatial consideration. In *European Conference on Computer Vision (ECCV)* (2004), pp. 342–354.

[KS02] KANG S. B., SHUM H.-Y.: A review of image-based rendering techniques. In *IEEE/SPIE Visual Communications and Image Processing 2000* (2002), pp. 2–13.

[KS04] KANG S. B., SZELISKI R.: Extracting view-dependent depth maps from a collection of images. *Int. J. Comput. Vision 58*, 2 (2004), 139–163.

[KSE*03] KWATRA V., SCHODL A., ESSA I., TURK G., BOBICK A.: Graphcut textures: Image and video synthesis using graph cuts. In *ACM SIGGRAPH* (2003), pp. 277–286.

[KSK06] KLAUS A., SORMANN M., KARNER K.: Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *International Conference on Pattern Recognition (ICPR)* (2006), pp. 15–18.

[KUWS03] KANG S. B., UYTTENDAELE M., WINDER S., SZELISKI R.: High dynamic range video. *ACM Transactions on Graphics 22*, 3 (July 2003), 319–325.

[Nis05] NISTER D.: Preemptive ransac for live structure and motion estimation. *Mach. Vision Appl. 16*, 5 (2005), 321–329.

[PAH*04] PETSCHNIGG G., AGRAWALA M., HOPPE H., SZELISKI R., COHEN M., TOYAMA K.: Digital photography with flash and no-flash image pairs. *ACM SIGGRAPH* (Aug. 2004), 664–672.

[PGB03] PEREZ P., GANGNET M., BLAKE A.: Poisson image editing. In *ACM SIGGRAPH* (2003), pp. 313–318.

[RAGS01] REINHARD E., ASHIKHMIN M., GOOCH B., SHIRLEY P.: Color transfer between images. *IEEE Computer Graphics and Applications 21*, 5 (2001), 34–41.

[RIY04] RASKAR R., ILIE A., YU J.: Image fusion for context enhancement and video surrealism. In *Non-Photorealistic Animation and Rendering (NPAR)* (2004), pp. 85–152.

[SCD*06] SEITZ S. M., CURLESS B., DIEBEL J., TEIN D. S., SZELISKI R.: A Comparison and Evaluation of Multi-View Stereo Reconstruction Algori thms. pp. 519–528.

[SCI05] SHECHTMAN E., CASPI Y., IRANI M.: Space-time super-resolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 27*, 4 (2005), 531–545.

[SGH*01] SAWHNEY H. S., GUO Y., HANNA K., KUMAR R., ADKINS S., ZHOU S.: Hybrid stereo camera: An IBR approach for synthesis of very high resolution stereoscopic image sequences. In *ACM SIGGRAPH* (2001), pp. 451–460.

[SK98] SEITZ S. M., KUTULAKOS K. N.: Plenoptic image editing. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision* (Washington, DC, USA, 1998), IEEE Computer Society, p. 17.

[Ska95] SKALICKY T.: *LASPack Reference Manual*, 1995.

[SS02] SCHARSTEIN D., SZELISKI R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision 47*, 1-3 (2002), 7–42.

[SSS06] SNAVELY N., SEITZ S. M., SZELISKI R.: Photo tourism: exploring photo collections in 3D. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (2006), pp. 835–846.

[ST04] SAND P., TELLER S.: Video matching. In *Proceedings of SIGGRAPH 2004* (2004), ACM Press / ACM SIGGRAPH, pp. 592–599.

[Sze06] SZELISKI R.: Locally adapted hierarchical basis preconditioning. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM Press, pp. 1135–1143.

[Tor98] TORR P.: Geometric motion segmentation and model selection, 1998.

[WB04] WILLS J., BELONGIE S.: A feature-based approach for determining long range optical flow. In *ECCV '04* (May 2004), pp. 170–182.

[WBC*05] WANG J., BHAT P., COLBURN A., AGRAWALA M., COHEN M.: Interactive video cutout. In *ACM SIGGRAPH* (2005), pp. 585–594.

[WJV*05] WILBURN B., JOSHI N., VAISH V., TALVALA E.-V., ANTUNEZ E., BARTH A., ADAMS A., HOROWITZ M., LEVOY M.: High performance imaging using large camera arrays. *ACM SIGGRAPH 24*, 3 (Aug. 2005), 765–776.

[WRA04] WANG H., RASKAR R., AHUJA N.: Seamless video editing. In *International Conference on Pattern Recognition* (2004).

[WSI04] WEXLER Y., SHECHTMAN E., IRANI M.: Space-time video completion. In *CVPR '04* (2004), pp. 120–127.

[YWY*06] YANG Q., WANG L., YANG R., STEWENIUS H., NISTER D.: Stereo matching with color-weighted correlation, hierachical belief propagation and occlusion handling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2006), pp. 2347–2354.

[ZK06] ZITNICK L., KANG S. B.: Stereo for image-based rendering using image over-segmentation. In *International Journal of Computer Vision* (2006).

[ZKU*04] ZITNICK C. L., KANG S. B., UYTTENDAELE M., WINDER S., SZELISKI R.: High-quality video view interpolation using a layered representation. *ACM Trans. Graph. 23*, 3 (2004), 600–608.