Space-Time Completion of Video

Yonatan Wexler, *Member*, *IEEE Computer Society*,
Eli Shechtman, *Student Member*, *IEEE Computer Society*, and
Michal Irani, *Member*, *IEEE Computer Society*

Abstract—This paper presents a new framework for the completion of missing information based on local structures. It poses the task of completion as a global optimization problem with a well-defined objective function and derives a new algorithm to optimize it. Missing values are constrained to form coherent structures with respect to reference examples. We apply this method to space-time completion of large space-time "holes" in video sequences of complex dynamic scenes. The missing portions are filled in by sampling spatio-temporal patches from the available parts of the video, while enforcing global spatio-temporal consistency between *all* patches in and around the hole. The consistent completion of static scene parts simultaneously with dynamic behaviors leads to realistic looking video sequences and images. Space-time video completion is useful for a variety of tasks, including, but not limited to: 1) Sophisticated video removal (of undesired static or dynamic objects) by completing the appropriate static or dynamic background information. 2) Correction of missing/corrupted video frames in old movies. 3) Modifying a visual story by replacing unwanted elements. 4) Creation of video textures by extending smaller ones. 5) Creation of complete field-of-view stabilized video. 6) As images are one-frame videos, we apply the method to this special case as well.

Index Terms—Video analysis, texture, space-time analysis.

1 Introduction

We present a method for *space-time completion* of large space-time "holes" in video sequences of complex dynamic scenes. We follow the spirit of [14] and use non-parametric sampling, while extending it to handle static and dynamic information simultaneously. The missing video portions are filled in by sampling spatio-temporal patches from other video portions, while enforcing global spatio-temporal consistency between all patches in and around the hole. Global consistency is obtained by posing the problem of video completion/synthesis as a global optimization problem with a *well-defined objective function* and solving it appropriately. The objective function states that the resulting completion should satisfy the following two constraints: 1) Every *local* space-time patch of the video sequence should be similar to some local space-time patch in the remaining parts of the video sequence (the "input data set"), while 2) *globally* all these patches must be consistent with each other, both spatially and temporally.

Solving the above optimization problem is not a simple task, especially due to the large dimensionality of video data. However, we exploit the spatio-temporal relations and redundancies to speed and constrain the optimization process in order to obtain realistic looking video sequences with complex scene dynamics at reasonable computation times.

- Y. Wexler is with Microsoft, 1 Microsoft Way, Redmond, WA 98052. E-mail: yonatan.wexler@microsoft.com.
- E. Shechtman and M. Irani are with the Department of Computer Science and Applied Math, Ziskind Building, The Weizmann Institute of Science, Rehovot, 76100 Israel. E-mail: {eli.shechtman, michal.irani}@weizmann.ac.il.

Manuscript received 19 July 2005; revised 15 Jan. 2006; accepted 27 June 2006; published online 15 Jan. 2007.
Recommended for acceptance by P. Torr.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0382-0705.

Fig. 1 shows an example of the task at hand. Given the input video (Fig. 1a), a space-time hole is specified in the sequence (Fig. 1b). The algorithm is requested to complete this hole using information from the remainder of the sequence. We assume that the hole is provided to the algorithm. While in all examples here it was marked manually, it can also be the outcome of some segmentation algorithm. The resulting completion and the output sequence are shown in Fig. 1c. The sources patches are shown in Fig. 2.

The goal of this work is close to a few well-studied domains. *Texture Synthesis* (e.g., [2], [14], [28]) extends and fills regular fronto-parallel image textures. This is similar to *Image Completion* (e.g., [9], [12]) which aims at filling in large missing image portions. While impressive results have been achieved recently in some very challenging cases (e.g., see [12]), the goal and the proposed algorithms have so far been defined only in a heuristic way. Global inconsistencies often result from independent local decisions taken at independent image positions. For this reason, these algorithms use large image patches in order to increase the chances of correct output. The two drawbacks of this approach are that elaborate methods for combining the large patches are needed for hiding inconsistencies [12], [13], [21], and that the data set needs to be artificially enlarged by including various skewed and scaled replicas that might be needed for completion. When compared to the pioneering work of [14] and its derivatives, the algorithm presented here can be viewed as stating an objective function explicitly. From this angle, the algorithm of [14] makes a greedy decision in each pixel based on the currently available pixels around it. It only uses a directional neighborhood around each pixel. A greedy approach requires the correct decision to be made at every step. Hence, the chances for errors increase rapidly as the gap grows. The work of [9] showed that this may be alleviated by prioritizing the completion order using local image structure. In challenging cases containing complex scenes and large gaps, the local neighborhood does not hold

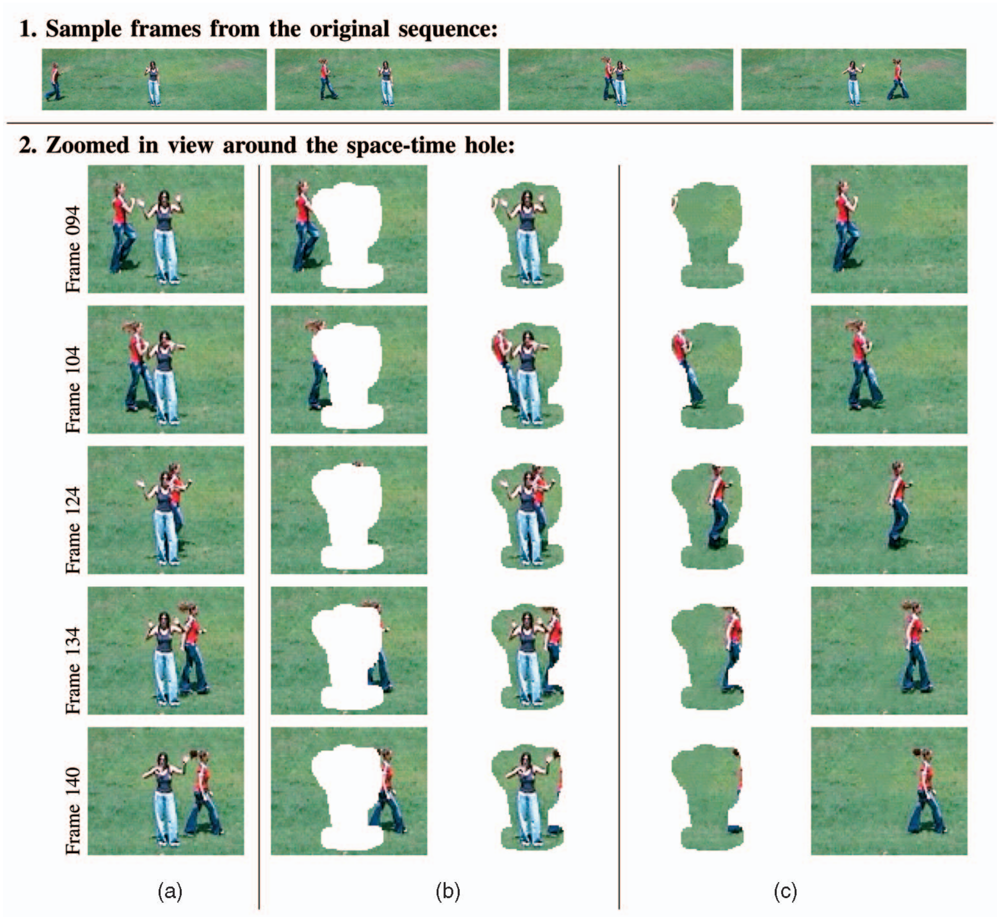
1. Sample frames from the original sequence:
2. Zoomed in view around the space-time hole:
Frame 094
Frame 104
Frame 124
Frame 134
Frame 140
(a)
(b)
(c)

Fig. 1. **1)** Few frames out of a video sequence showing one person standing and waving her hands while the other person is hopping behind her. The video size is $100 \times 300 \times 240$, with 97,520 missing pixels. **2)** A zoomed-in view on a portion of the video around the space-time hole before and after the completion. Note that the recovered arms of the hopping person are at slightly different orientations than the removed ones. As this particular instance does not exist anywhere else in the sequence, a similar one from a different time instance was chosen to provide an equally likely completion. See video at: http://www.wisdom.weizmann.ac.il/~vision/VideoCompletion.html. (a) Input Sequence. (b) Occluder cut out manually. (c) Spatio-temporal completion.

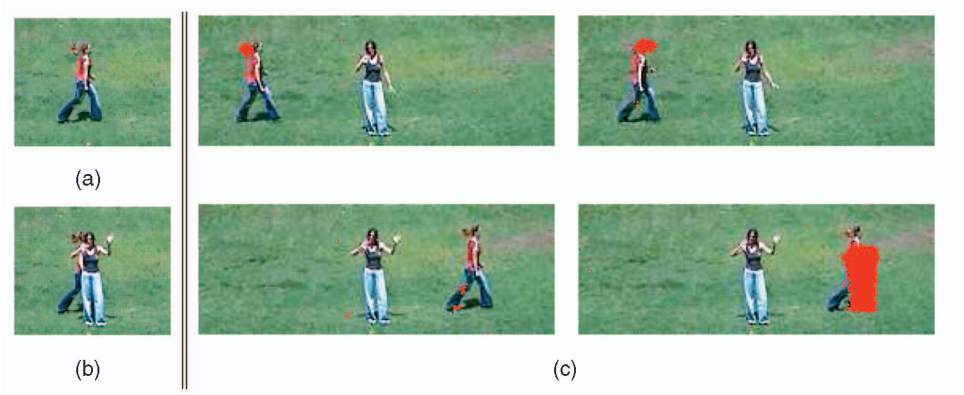

Fig. 2. Sources of information. Output frame 114, shown in (a), is a combination of the patches marked here in red over the input sequence (c). It is noticeable that large continuous regions have been automatically picked whenever possible. Note that the hair, head, and body were taken from different frames. The original frame is shown in (b). (a) Output. (b) Input. (c) Selected patches.

enough information for a globally correct solution. This is more pronounced in video. Due to motion aliasing, there is little chance that an exact match will be found.

The framework presented here requires that the *whole* neighborhood around each pixel is considered, not just a causal subset of it. Moreover, it considers *all* windows containing each pixel simultaneously, thus effectively using an even larger neighborhood.

Image Inpainting (e.g., [5], [6], [22]) was defined in a principled way as an edge continuation process, but is restricted to small (narrow) missing image portions in highly structured image data. These approaches have been restricted to the completion of *spatial information* alone in images. Even when applied to video sequences (as in [4]), the completion was still performed spatially. The temporal component of video has mostly been ignored. The basic assumption of Image Inpainting, that edges should be interpolated in some smooth way, does not naturally extend to time. Temporal aliasing is typically much stronger than spatial aliasing in video sequences of dynamic scenes. Often, a pixel may contain background information in one frame and foreground information in the next frame, resulting in very nonsmooth temporal changes. These violate the underlying assumptions of Inpainting.

In [19], a method has been proposed for employing spatio-temporal information to correct scratches and noise in poor-quality video sequences. This approach relies on optical-flow estimation and propagation into the missing parts followed by reconstruction of the color data. The method was extended to removal of large objects in [20] under the assumption of planar rigid layers and small camera motions.

View Synthesis deals with computing the appearance of a scene from a new viewpoint, given several images. A similar objective was already used in [15] for successfully resolving ambiguities which are otherwise inherent to the geometric problem of new-view synthesis from multiple camera views. The objective function of [15] was defined on 2D images. Their local distance between 2D patches was based on SSD of color information and included geometric constraints. The algorithm there did not take into account the dependencies between neighboring pixels as only the central point was updated in each step.

Recently, there have been a few notable approaches to video completion. From the algorithmic perspective, the most similar work to ours is [7], which learns a mapping from the input video into a smaller volume, aptly called "epitome." These are then used for various tasks including completion. While our work has a similar formulation, there are major differences. We seek some cover of the missing data by the available information. Some regions may contribute more than once while some may not contribute at all. In contrast, the epitome contains a proportional representation of all the data. One implication of this is that the windows will be averaged and, so, will lose some fidelity. The recent work of [24] uses estimated optical flow to separate foreground and background layers. Each is then filled incrementally using the priority-based ordering idea similar to [9].

Finally, the work of [18] takes an object-based approach where large portions of the video are tracked, their cycles are analyzed, and they can then be inserted into the video. This allows warping of the object so it fits better, but requires a complete appearance of the object to be identified and, so, is not applicable to more complex dynamics, such as articulated motion or stochastic textures.

A closely related area of research which regards temporal information explicitly is that of *dynamic texture synthesis* in videos (e.g., [3], [11]). Dynamic textures are usually characterized by an unstructured stochastic process. They model and synthesize smoke, water, fire, etc., but cannot model nor synthesize structured dynamic objects, such as behaving people. We demonstrate the use of our method for synthesizing large video textures from small ones in Section 6. While [26] has been able to synthesize/complete video frames of structured dynamic scenes, it assumes that the "missing" frames already appear in their entirety elsewhere in the input video and, therefore, needed only to identify the correct permutation of frames. An extension of that paper [25] manually composed smaller "video sprites" to a new sequence.

The approach presented here can automatically handle the completion and synthesis of both structured dynamic objects as well as unstructured dynamic objects under a single framework. It can complete frames (or portions of them) that never existed in the data set. Such frames are constructed from various space-time patches, which are automatically selected from different parts of the video sequence, all put together consistently. The use of a global objective function removes the pitfalls of local inconsistencies and the heuristics of using large patches. As can be seen in the figures and in the attached videos, the method is capable of completing large space-time areas of missing information containing complex structured dynamic scenes, just as it can work on complex images. Moreover, this method provides a unified framework for various types of image and video completion and synthesis tasks, with the appropriate choice of the spatial and temporal extents of the space-time "hole" and of the space-time patches. A preliminary version of this work appeared in CVPR '04 [29].

This paper is organized as follows: Section 2 introduces the objective function and Section 3 describes the algorithm used for optimizing it. Sections 4 and 5 discuss trade-offs between space and time dimensions in video and how they are unified into one framework. Section 6 demonstrates the application of this method to various problems. Finally, Section 7 concludes this paper.

2 Completion as a Global Optimization

Given a video sequence, we wish to complete the missing portions in such a way that it looks just like the available parts. For this, we define a global objective function to rank the quality of a completion. Such a function needs to take a completed video and rate its quality with respect to a reference one. Its extremum should denote the best possible solution.

Our basic observation is that, in order for a video to look good, it needs to be coherent everywhere. That is, a good completion should resemble the given data locally everywhere.

The constraint on the color of each pixel depends on the joint color assignment of its neighbors. This induces a structure where each pixel depends on the neighboring ones. The correctness of a pixel value depends on whether its local neighborhood forms a coherent structure. Rather than modeling this structure, we follow [14] and use the reference video as a library of video samples that are considered to be coherent.

Given these guidelines, we are now ready to describe our framework in detail.

2.1 The Global Objective Function

To allow for a uniform treatment of dynamic and static information, we treat video sequences as space-time volumes. We use the following notations: A pixel (x, y) in a frame t will be regarded as a space-time point $p = (x, y, t)$ in the volume. W_p denotes a small, fixed-sized window around the point p both in space and in time. The diameter of the window is given as a parameter. We use indices i and j to denote locations relative to p. For example, W_p is the window centered around p and W_p^i is the $i\text{th}$ window containing p and is centered around the $i\text{th}$ neighbor of p.

We say that a video sequence $\mathcal{S}$ has *global visual coherence* with some other sequence $\mathcal{D}$ if every local space-time patch in

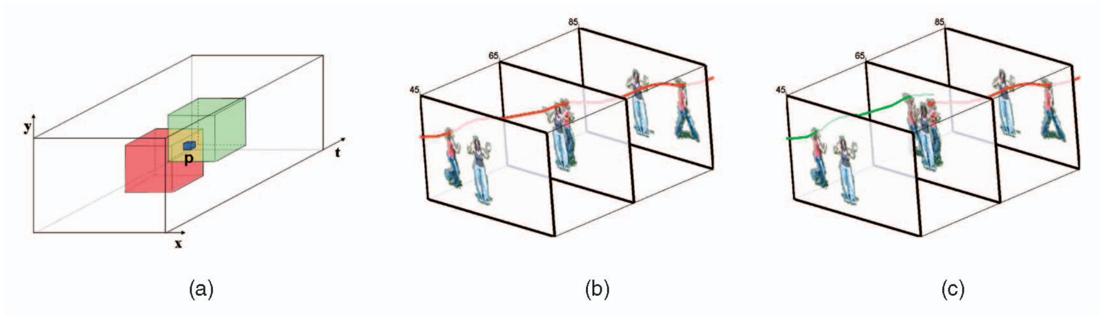

Fig. 3. Local and global space-time consistency. (a) Enforcement of the global objective function of (1) requires coherence of all space-time patches containing the point p. (b) Such coherence leads to a globally correct solution. The true trajectory of the moving object is marked in red and is correctly recovered. When only local constraints are used and no global consistency enforced, the resulting completion leads to inconsistencies (c). The background figure is wrongly recovered twice with wrong motion trajectories.

$\mathcal{S}$ can be found somewhere within the sequence $\mathcal{D}$. In other words, we can cover $\mathcal{S}$ with small windows from $\mathcal{D}$. Windows in the data set $\mathcal{D}$ are denoted by V and are indexed by a reference pixel (e.g., V_q and V_q^i).

Let $\mathcal{S}$ and $\mathcal{H} \subseteq \mathcal{S}$ be an input sequence and a "hole" region within it. That is, $\mathcal{H}$ denotes all the missing space-time points within $\mathcal{S}$. For example, $\mathcal{H}$ can be an undesired object to be erased, a scratch or noise in old corrupt footage, or entire missing frames, etc. We assume that both $\mathcal{S}$ and $\mathcal{H}$ are given.

We wish to complete the missing space-time region $\mathcal{H}$ with some new data $\mathcal{H}^*$ such that the resulting video sequence $\mathcal{S}^*$ will have as much global visual coherence with some reference sequence $\mathcal{D}$ (the data set). Typically, $\mathcal{D} = \mathcal{S} \setminus \mathcal{H}$, namely, the remaining video portions outside the hole, are used to fill in the hole. Therefore, we seek a sequence $\mathcal{S}^*$ which maximizes the following objective function:

$$\text{Coherence}(\mathcal{S}^*|\mathcal{D}) = \prod_{p \in \mathcal{S}^*} \max_{q \in \mathcal{D}} sim(W_p, V_q), \qquad (1)$$

where p, q run over all space-time points in their respective sequences. $sim(\cdot, \cdot)$ is a local similarity measure between two space-time patches that will be defined shortly (Section 2.2). The patches need not necessarily be isotropic and can have different sizes in the spatial and temporal dimensions. We typically use $5 \times 5 \times 5$ patches which are large enough to be statistically meaningful but small enough so that effects, such as parallax or small rotations, will not affect them. They are the basic building blocks of the algorithm. The use of windows with temporal extent assumes that the patches are correlated in time, and not only in space. When the camera is static or fully stabilized, this is trivially true. However, it may also apply in other cases where the same camera motion appears in the database (as shown in Fig. 16).

Fig. 3 explains why (1) induces global coherence. Each space-time point p belongs to other space-time patches of other space-time points in its vicinity. For example, for a $5 \times 5 \times 5$ window, 125 different patches involve p. The red and green boxes in Fig. 3a are examples of two such patches. Equation (1) requires that all 125 patches agree on the value of p and, therefore, (1) leads to globally coherent completions such as the one in Fig. 3b. If the global coherence of (1) is not enforced, and the value of p is determined locally by a single best-matching patch (i.e., using a sequential greedy algorithm as in [9], [14], [24]), then global inconsistencies will occur in a later stage of the recovery. An example of temporal incoherence is shown in Fig. 3c. A greedy approach requires the correct answer to be found at every step and this is rarely the case as local image structure will often contain ambiguous information.

The above objective function seeks a cover of the missing data using the available one. That is, among the exponentially large number of possible covers, we seek the one that will give us the least amount of total error. The motivation here is to find a solution that is correct everywhere, as any local inconsistency will push the solution away from the minimum.

2.2 The Local Space-Time Similarity Measure

At the heart of the algorithm is a well-suited similarity measure between space-time patches. A good measure needs to agree *perceptually* with a human observer. The Sum of Squared Differences (SSD) of color information, which is so widely used for image completion, does not suffice for the desired results in video (regardless of the choice of color space). The main reason for this is that the human eye is very sensitive to motion. Maintaining motion continuity is more important than finding the exact spatial pattern match within an image of the video.

Fig. 4 illustrates (in 1D) that very different temporal behaviors can lead to the same SSD score. The function $f(t)$ has a noticeable temporal change. Yet, its SSD score relative to a similar looking function $g_1(t)$ is the same as the SSD score of $f(t)$ with a flat function $g_2(t)$: $\int(f - g_1)^2 dt = \int(f - g_2)^2 dt$. However, perceptually, $f(t)$ and $g_1(t)$ are more similar, as they both encode a temporal change.

We would like to incorporate this into our algorithm so to create a similarity measure that agrees with perceptual similarity. Therefore, we add a measure that is similar to that of normal-flow to obtain a quick and rough approximation of the motion information. Let Y be the sequence containing the grayscale (intensity) information obtained from the color sequence. At each space-time point, we compute the spatial and temporal derivatives (Y_x, Y_y, Y_t). If the motion were only horizontal, then $u = Y_t/Y_x$ would capture the instantaneous motion in the x direction. If the

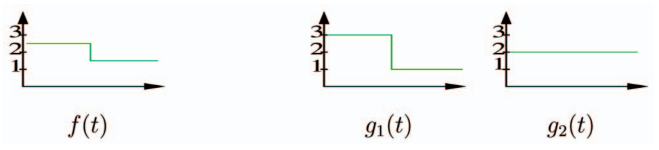

Fig. 4. Importance of matching derivatives (see text).

motion were only vertical, then $v = Y_t/Y_y$ would capture the instantaneous motion in the y direction. The fractions factor out most of the dependence between the spatial and the temporal changes (such as frame-rate) while capturing object velocities and directions. These two components are scaled and added to the RGB measurements to obtain a five-dimensional representation for each space-time point: $(R, G, B, \alpha u, \alpha v)$, where $\alpha = 5$. Note that we do not compute optical flow. We apply an L_2-norm (SSD) to this 5D representation in order to capture space-time similarities for static and dynamic parts simultaneously. Namely, for two space-time windows W_p and V_q, we have $d(W_p, V_q) = \sum_{(x,y,t)} \|W_p(x,y,t) - V_q(x,y,t)\|^2$, where, for each (x,y,t) within the patch, $W_p(x,y,t)$ denotes its 5D measurement vector. The distance is translated to the similarity measure

$$sim(W_p, V_q) = e^{-\frac{d(W_p,V_q)}{2\sigma^2}}. \tag{2}$$

The choice of σ is important as it controls the smoothness of the induced error surface. Rather than using a fixed value, it is chosen to be the 75-percentile of all distances in the current search in all locations. In this way, the majority of locations are taken into account and, hence, there is a high probability that the overall error will reduce.

3 The Optimization

The inputs to the optimization are a sequence $\mathcal{S}$ and a "hole" $\mathcal{H} \subset \mathcal{S}$, marking the missing space-time points to be corrected or filled in. The algorithm seeks an assignment of color values for all the space-time points (pixels) in the hole so to satisfy (1). While (1) does not imply any optimization scheme, we note that it will be satisfied if the following two local conditions are met at every space-time point p:

1. All windows $W_p^1 \dots W_p^k$ containing p appear in the data set $\mathcal{D}$:
$$\exists V^i \in \mathcal{D},\ W_p^i = V^i.$$
2. All those $V^1 \dots V^k$ agree on the color value c at location p:
$$c = V^i(p) = V^j(p).$$

This is trivially true since the second condition is a particular case of the first. These conditions imply an iterative algorithm. The iterative step will aim at satisfying these two conditions locally at every point p. Any change in p will affect all the windows that contain it and, so, the update rule must take all of them into account.

Note that (1) may have an almost trivial solution in which the hole $\mathcal{H}$ contains an exact copy of some part in the database. For such a solution, the error inside the hole will be zero (as both conditions above are satisfied) and, so, the total coherence error will be proportional to the surface area of the space-time boundary. This error might be much smaller than small noise errors spread across the entire volume of the hole. Therefore, we associate an additional quantity α_p to each point $p \in \mathcal{S}$. Known points $p \in \mathcal{S} \setminus \mathcal{H}$ have fixed high confidence, whereas missing points $p \in \mathcal{H}$ will have lower confidence. This weight is chosen so to ensure that the total error inside the hole is less than that on the hole boundary. This argument needs to hold recursively in every layer in the hole. An approximation to such weighting is to compute the distance transform for every hole pixel and then use $\alpha_p = \gamma^{-\text{dist}}$. When the hole is roughly spherical, choosing $\gamma = 1.3$ gives the desired weighting. This measure bears some similarity to the priority used in [9] except that, here, it is fixed throughout the process. Another motivation for using such weighting is to speed up convergence by directing the flow of information inward from the boundary.

3.1 The Iterative Step

Let $p \in \mathcal{H}$ be some hole point which we wish to improve. Let $W_p^1 \dots W_p^k$ be all space-time patches containing p. Let $V^1 \dots V^k$ denote the patches in $\mathcal{D}$ that are most similar to $W_p^1 \dots W_p^k$ per (2). According to Condition 1 above, if W_p^i is reliable, then $d(W_p^i, V^i) \approx 0$. Therefore, $sim(W_p^i, V^i)$ measures the degree of reliability of the patch W_p^i.

We need to pick a color c at location p so that the coherence at all windows containing it will increase. Each window V^i provides an evidence of possible solution for c and the confidence in this evidence is given by (2) as $s_p^i = sim(W_p^i, V^i)$. According to Condition 2 above, the most likely color c at p should minimize the variance of the colors $c^1 \dots c^k$ proposed by $V^1 \dots V^k$ at location p. As mentioned before, these values should be scaled according to the fixed α_p^i to avoid the trivial solution. Thus, the most likely color at p will minimize $\sum_i \omega_p^i (c - c^i)^2$, where $\omega_p^i = \alpha_p^i \cdot s_p^i$. Therefore,

$$c = \frac{\sum_i \omega_p^i c^i}{\sum_i \omega_p^i}. \tag{3}$$

This c is assigned to be the color of p at the end of the current iteration. Such an update rule minimizes the local error around each space-time point $p \in \mathcal{H}$ while maintaining consistency in all directions, leading to a global consistency.

One drawback of the update rule in (3) is its sensitivity to outliers. It is enough that a few neighbors suggest the wrong color to bias the mean color c and thus prevent or delay the convergence of the algorithm. In order to avoid such effects, we treat the k possible assignments as evidence. These evidences give discrete samples in the continuous space of possible color assignments. The reliability of each evidence is proportional to ω_p^i and we seek the Maximum Likelihood (ML) in this space. This is computed using the Mean-Shift algorithm [8] with a variable bandwidth (window size). The Mean-Shift algorithm finds the density modes of the distribution (which is related to Parzen windows density estimation). It is used here to extract the dominant mode of the density. The bandwidth is defined with respect to the standard deviation σ of the colors $c_1, \dots, c_k$ at each point. It is typically set to be 3σ at the beginning and is reduced gradually to 0.2σ. The highest mode M is picked and so the update rule in Fig. 5 is:

$$c = \frac{\sum_{i \in \mathrm{M}} \omega_p^i c^i}{\sum_{i \in \mathrm{M}} \omega_p^i}. \tag{4}$$

This update rule produces a robust estimate for the pixel in the presence of noise. While making the algorithm slightly more complex, this step has further advantages. The bandwidth parameter controls the degree of smoothness of the error surface. When it is large, it reduces to simple weighted averaging (as in (3)), hence allowing rapid convergence.

```
1: Input: video $\mathcal{S}$, hole $\mathcal{H} \subset \mathcal{S}$, database $\mathcal{D}$.
2: Compute space-time pyramids $\mathcal{S}_l$, $\mathcal{H}_l$, $\mathcal{D}_l$ $l = 1..L$.
3: $t \leftarrow 0$,  $\mathcal{S}^t \leftarrow \mathcal{S}$
4: for pyramid level $l$, from top to bottom do
5:   repeat
6:     for all $p \in \mathcal{H}_l$ do
7:       Let $\{W_p^i\}_{i=1}^k$ be all windows s.t. $p \in W_p^i$
8:       Find $\{V^i\} \subseteq \mathcal{D}_l$ maximizing Eq. (2)
9:       Let $c^i \in V^i$ be the appropriate colors.
10:      Set $\omega_p^i = \alpha_p^i \cdot sim(W_p^i, V^i)$.
11:      $\mathcal{S}^{t+1}(p) \leftarrow ML(c^i, \omega_p^i)$ using Eq. (4)
12:    end for
13:    $t \leftarrow t+1$
14:  until convergence
15:  Propagate solution to the next level
16: end for
17: Output: $\mathcal{S}^t$
```

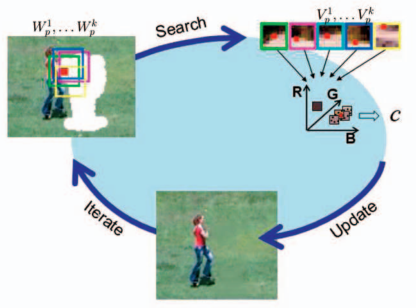

Fig. 5. Video completion algorithm.

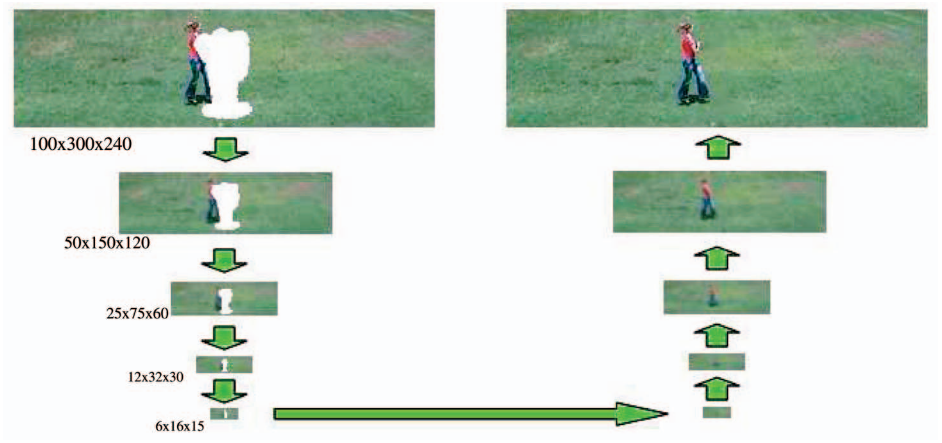

Fig. 6. Multiscale solution. The algorithm starts by shrinking the input video (both in space and in time). The completion starts at the coarsest level, iterating each level several times and propagating the result upward.

When it is small, it induces a weighted majority vote, avoiding blurring in the resulting output. The use of a varying σ value has significantly improved the convergence of the algorithm. When compared with the original work in [29], the results shown here achieve better convergence. This can be seen from the improved amount of detail in areas which were previously smoothed unnecessarily. The ability to rely on outlier rejection also allows us to use approximate nearest neighbors, as discussed in Section 3.3. The algorithm is summarized in Fig. 5 both graphically and in pseudocode.

3.2 Multiscale Solution

To further enforce global consistency and to speed up convergence, we perform the iterative process in multiple scales using spatio-temporal pyramids. Each pyramid level contains half the resolution in the spatial and in the temporal dimensions. The optimization starts at the coarsest pyramid level and the solution is propagated to finer levels for further refinement. Fig. 6 shows a typical multiscale V-cycle performed by the algorithm. It is worth mentioning that, as each level contains 1/8th of the pixels, both in the hole $\mathcal{H}$ and in the database $\mathcal{D}$, the computational cost of using a pyramid is almost negligible (8/7 of the work). In hard examples, it is sometimes necessary to repeat several such cycles, gradually reducing the pyramid height. This is inspired by multigrid methods [27].

The propagation of the solution from a coarse level to the one above it is done as follows: Let $p_\uparrow$ be a location in the finer level and let $p_\downarrow$ be its corresponding location in the coarser level. As before, let Let $W_{p_\downarrow}^i$ be the windows around $p_\downarrow$ and let $V_\downarrow^i$ be the matching windows in the database. We propagate the locations of $V_\downarrow^i$ onto the finer level to get $V_\uparrow^i$. Some of these (about $\frac{k}{8}$, half in each dimension) will overlap $p_\uparrow$ and these will be used for the maximum-likelihood step, just as before (except that here there are less windows). This method is better than plain interpolation as the initial guess for the next level will preserve high spatio-temporal frequencies and will not be blurred unnecessarily.

3.3 Algorithm Complexity

We now discuss the computational complexity of the suggested method. Assume we have $N = |\mathcal{H}|$ pixels in the

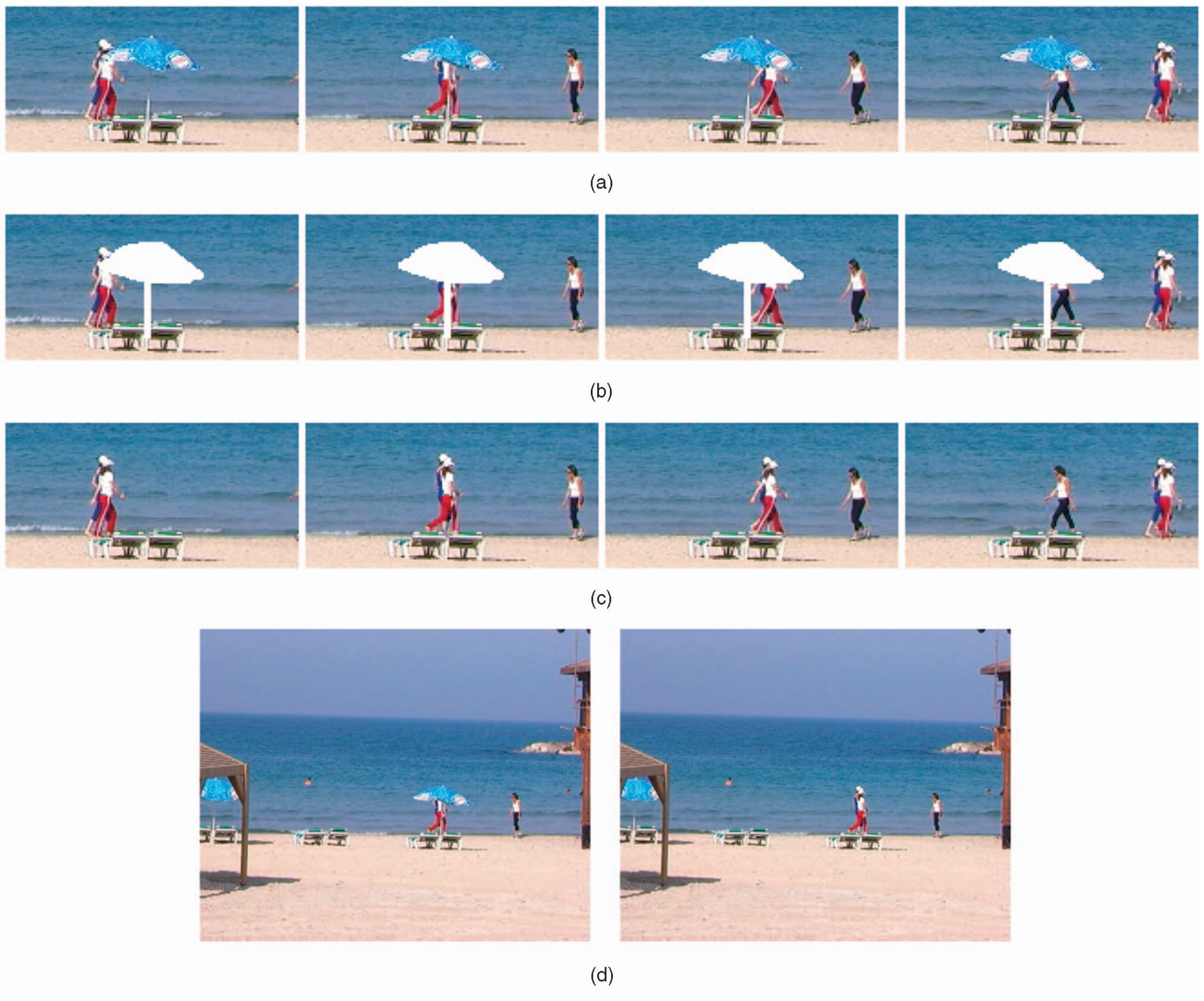

Fig. 7. Removal of a beach umbrella. The video size is $120 \times 340 \times 100$, with 422,000 missing pixels. See the video at: http://www.wisdom.weizmann.ac.il/~vision/VideoCompletion.html. (a) Input sequence. (b) Umbrella removed. (c) Output. (d) Full-size frames (input and output).

hole and that there are K windows in the database. The algorithm has the following structure.

First, a spatio-temporal pyramid is constructed and the algorithm iterates a few times in each level (typically, five to 10 iterations in each level). Level l contains roughly $N/8^l$ hole pixels and $K/8^l$ database windows.

Each iteration of the algorithm has two stages: searching the database and per-pixel Maximum Likelihood.

The first stage performs a nearest-neighbor search once for each window overlapping the space-time hole $\mathcal{H}$. This is the same computational cost as all the derivatives from Efros and Leung's work [14]. The search time depends on K and the nearest-neighbor search method. Since each patch is searched independently, this step can be parallelized trivially. While brute-force would take $O(K/8^l \cdot N/8^l)$, we use the method of [1] so the search time is logarithmic in K. We typically obtain a speedup of two orders of magnitude over brute force search. This approach is very suitable for our method for three reasons: First, it is much faster. Second, we always search for windows of the same size. Third, our robust maximum-likelihood estimation allows us to deal with wrong results that may be returned by this approximate algorithm.

The second stage is the per pixel maximum-likelihood computation. We do this using the Mean-Shift algorithm [8]. The input is a set of 125 RGB triplets along with their weights. A trivial implementation is quadratic in this small number and so is very fast.

The running time depends on the above factors. For small problems, such as image completion, the running time is about a minute. For the "Umbrella" sequence in Fig. 7 of size $120 \times 340 \times 100$ with 422,000 missing pixels each iteration in the top level takes roughly one hour on a 2.6Ghz Pentium computer. Roughly 95 percent of this time is used for the nearest neighbor search. This suggests that pruning the database (as in [24]) or simplifying the search (as in [2]) would give a significant speedup.

3.4 Relation to Statistical Estimation

The above algorithm can be viewed in terms of a statistical estimation framework. The global visual coherence in (1) can be derived as a Likelihood function via a graphical model, and our iterative optimization process can be seen as an approximation of the EM method to find the maximum-likelihood estimate. According to this model, the parameters are the colors of the missing points in the space-time hole and pixels in the boundary around the hole are the observed variables. The space-time patches are the hidden variables and are drawn from the data set (patches outside the hole in our case).

We show in the Appendix how the likelihood function is derived from this model, and that the maximum-likelihood solution is the best visually coherent completion, as defined in (1). We also show that our optimization algorithm fits the EM method [10] for maximizing the Likelihood function. Under some simplifying assumptions, the $\mathbf{E}$ step is equivalent to the nearest neighbor match of a patch from the data set to the current estimated corresponding "hole" patch. The $\mathbf{M}$ step is equivalent to the update rule of (3).

The above procedure also bears some similarity to the Belief Propagation (BP) approach to completion such as in [22]. As BP communicates a PDF (probability density function), it is practically limited to modeling no more than three neighboring connections at once. There, a standard grid-based graphical model is used (as is, for example, in [16]), in which each pixel is connected to its immediate neighbors. Our graphical model has a *completely different* structure. Unlike BP, our model does not have links between nodes (pixels). Rather, the patch structure induces an implicit connection between nodes. It takes into account not only a few immediate neighbors, but *all* of them (e.g., 125). This allows us to deal with more complex visual structures. In addition, we assume that the point values are parameters and not random variables. Instead of modeling the PDF, we use the local structure to estimate its likelihood and derive an update step using the nearest neighbor in the data set.

When seen as a variant of belief propagation, one can think of this method as propagating only one evidence. In the context of this work, this is reasonable, as the space of all patches is very high dimensional and is sparsely populated. Typically, the data set occupies "only" a few million samples out of $(3 * 255)^{125}$ possible combinations. The typical distance between the points is very high and, so, only a few are relevant, so passing a full PDF is not advantageous.

4 Space-Time Visual Trade-offs

The spatial and temporal dimensions are very different in nature, yet are interrelated. This introduces visual trade-offs between space and time that are beneficial to our space-time completion process. On one hand, these relations are exploited to narrow down the search space and to speed up the completion process. On the other hand, they often entail different treatments of the spatial and temporal dimensions in the completion process. Some of these issues have been mentioned in previous sections in different contexts, and are therefore only briefly mentioned here. Other issues are discussed here in more length.

Temporal versus spatial aliasing. Typically, there is more temporal aliasing than spatial aliasing in video sequences of dynamic scenes. This is mainly due to the different nature of blur functions that precede the sampling process (digitization) in the spatial and in the temporal dimensions: The spatial blur induced by the video camera (a Gaussian whose extent is several pixels) is a much better low-pass filter than the temporal blur induced by the exposure time of the camera (a Rectangular blur function whose extent is less than a single frame-gap in time). This leads to a number of observations:

1. Extending the family of Inpainting methods to include the temporal dimension may be able to handle completion of (narrow) missing video portions that undergo slow motions, but there is a high likelihood that it will not be able to handle fast motions or even simple everyday human motions (such as walking, running, etc). This is because Inpainting relies on edge continuity, which will be hampered by strong temporal aliasing.

   Space-time completion, on the other hand, does not rely on smoothness of information within patches and can therefore handle aliased data as well.
2. Because temporal aliasing is shorter than spatial aliasing, our multiscale treatment is not identical in space and in time. In particular, after applying the video completion algorithm of Section 3, residual spatial (appearance) errors may still appear in fast recovered moving objects. To correct those effects, an additional refinement step of space-time completion is added, but this time, only the spatial scales vary (using a spatial pyramid), while the temporal scale is kept at the original temporal resolution. The completion process, however, is still space-time. This allows for completion using patches which have a large spatial extent to correct the spatial information, while maintaining a minimal temporal extent so that temporal coherence is preserved without being affected too much by the temporal aliasing.

The local patch size. In our space-time completion process, we typically use $5 \times 5 \times 5$ patches. Such a patch size provides $5^3 = 125$ measurements per patch. This usually provides sufficient statistical information to make reliable inferences based on this patch. To obtain a similar number of measurements for reliable inference in the case of 2D image completion, we would need to use patches of size 11×11. Such patches, however, are not small, and are therefore more sensitive to geometric distortions (effects of 3D parallax, change in scale, and orientation) due to different viewing directions between the camera and the imaged objects. This restricts the applicability of image-based completion or else requires the use of patches at different sizes and orientations [12], which increases the complexity of the search space combinatorially.

One may claim that, due to the new added dimension (time), there is a need to select patches with a larger number of samples to reflect the increase in data complexity. This, however, is not the case, due to the large degree of spatio-temporal redundancy in video data. The data complexity indeed increases slightly, but this increase is in no way proportional to the increase in the amounts of data.

The added temporal dimension therefore provides greater flexibility. The locality of the $5 \times 5 \times 5$ patches both in space and in time makes them relatively insensitive to small variations in scaling, orientation, or viewing direction and, therefore, applicable to a richer set of scenes (richer both in the spatial sense and in the temporal sense).

Interplay between time and space. Often, the lack of spatial information can be compensated for by the existence of

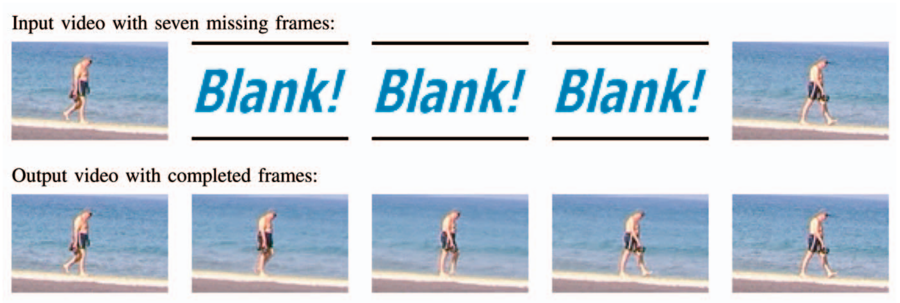
Input video with seven missing frames:
Blank!
Blank!
Blank!
Output video with completed frames:

Fig. 8. Completion of missing frames. The video size is $63 \times 131 \times 42$ and has 57,771 missing pixels in seven frames.

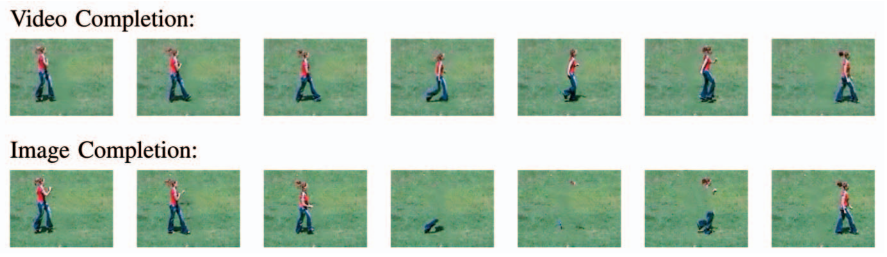

Fig. 9. Image completion versus video completion.

temporal information, and vice versa. To show the importance of combining the two cues of information, we compare the results of spatial completion alone to those of space-time completion. The top row of Fig. 9 displays the resulting completed frames of Fig. 1 using space-time completion. The bottom row of Fig. 9 shows the results obtained by filling in the same missing regions, but this time, using only image (spatial) completion. In order to provide the 2D image completion with the best possible conditions, the image completion process was allowed to choose the spatial image patches from *any* of the frames in the input sequence. It is clear from the comparison that image completion failed to recover the dynamic information. Moreover, it failed to complete the hopping woman in any reasonable way, regardless of the temporal coherence.

Furthermore, due to the large spatio-temporal redundancy in video data, the added temporal component provides additional flexibility in the completion process. When the missing space-time region (the "hole") is spatially large and temporally small, then the temporal information will provide most of the constraints in the completion process. In such cases, image completion will not work, especially if the missing information is dynamic. Similarly, if the hole is temporally large, but spatially small, then spatial information will provide most of the constraints in the completion, whereas pure temporal completion/synthesis will fail. Our approach provides a unified treatment of all these cases, without the need to commit to a spatial or a temporal treatment in advance.

5 Unified Approach to Completion

The approach presented in this paper provides a unified framework for various types of image and video completion/synthesis tasks. With the appropriate choice of the spatial and temporal extents of the space-time "hole" $\mathcal{H}$ and of the space-time patches W_p, our method reduces to any of the following special cases:

1. When the space-time patches W_p of (1) have only a spatial extent (i.e., their temporal extent is set to 1), then our method becomes the classical *spatial* image completion and synthesis. However, because our completion process employs a global objective function (1), global consistency is obtained that is otherwise lacking when not enforced. A comparison of our method to other image completion/synthesis methods is shown in Figs. 12, 13, and 14. (We could not check the performance of [12] on these examples. We have, however, applied our method to the examples shown in [12] and obtained comparably good results.)
2. When the spatial extent of the space-time patches W_p of (1) is set to be the entire image, then our method reduces to *temporal* completion of missing frames or synthesis of new frames using existing frames to fill in temporal gaps (similar to the problem posed by [26]).
3. If, on the other hand, the spatial extent of the space-time "hole" $\mathcal{H}$ is set to be the entire frame (but the patches W_p of (1) remain small), then our method still reduces to *temporal* completion of missing video frames (or synthesis of new frames), but this time, unlike [26], the completed frames may have never appeared in their entirety anywhere in the input sequence. Such an example is shown in Fig. 8, where three frames were dropped from the video sequence of a man walking on the beach. The completed frames were synthesized from bits of information gathered from different portions of the remaining video sequence. Waves, body, legs, arms, etc., were automatically selected from different space-time locations in the sequence so that they all match coherently (both in space and in time) to each other as well as to the surrounding frames.

6 Applications

Space-time video completion is useful for a variety of tasks in video postproduction and video restoration. A few

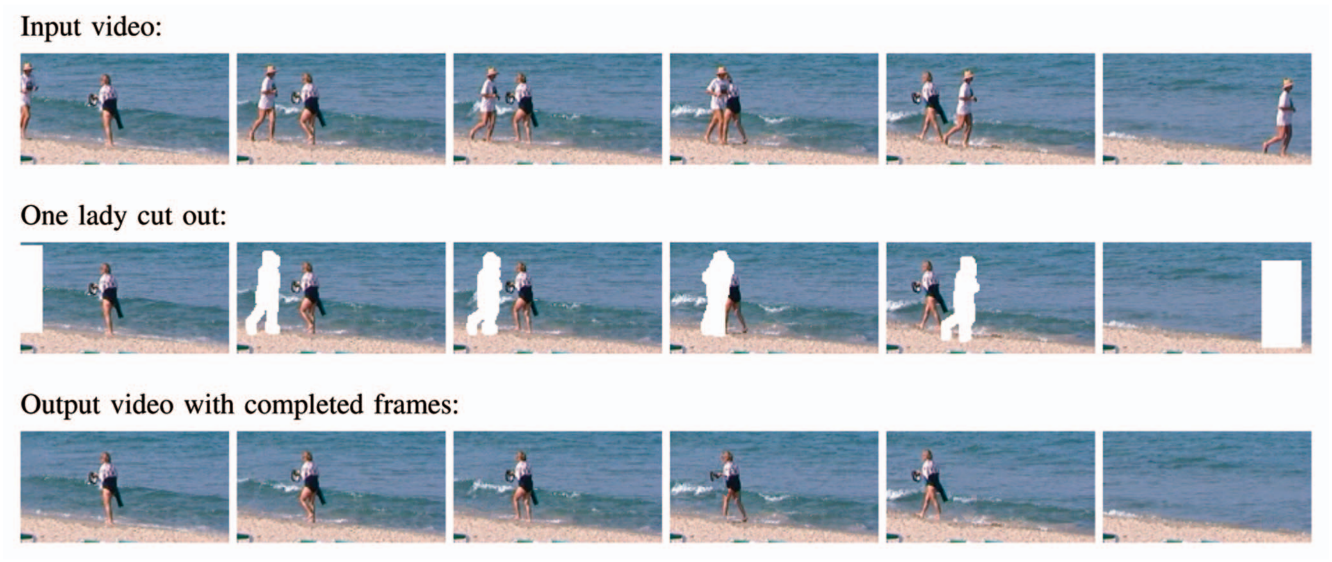

Fig. 10. Removal of a running person. The video size is $80 \times 170 \times 88$ with 74,625 missing pixels.

example applications are listed below. In all the examples below, we crop the video so to contain only relevant portions. The size of the original videos for most of them is half PAL video resolution (360×288).

1. **Sophisticated video removal**. Video sequences often contain undesired objects (static or dynamic), which were either not noticed or else were unavoidable at the time of recording. When a moving object reveals all portions of the background at different time instances, then it can be easily removed from the video data, while the background information can be correctly recovered using simple techniques (e.g., [17]). Our approach can handle the more complicated case, when portions of the background scene are never revealed, and these occluded portions may further change dynamically. Such examples are shown in Figs. 1, 7, and 10. Note that, in both Figs. 7 and 10, all the completed information is dynamic and, so, reliance on background subtraction or segmentation is not likely to succeed here.
2. **Restoration of old movies**. Old video footage is often very noisy and visually corrupted. Entire frames or portions of frames may be missing, and severe noise may appear in other video portions. These kinds of problems can be handled by our method. Such an example is shown in Fig. 11.

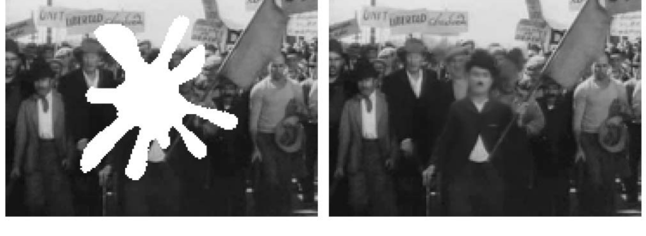

Fig. 11. Restoration of a corrupted old Charlie Chaplin movie. The video is of size $135 \times 180 \times 96$ with 3,522 missing pixels.

3. **Modify a visual story**. Our method can be used to make people in a movie change their behavior. For example, if an actor has absent mindedly picked his nose during a film recording, then the video parts containing the obscene behavior can be removed, to be coherently replaced by information from data containing a range of "acceptable" behaviors. This is demonstrated in Fig. 15, where an unwanted scratching of the ear is removed.
4. **Complete field-of-view of a stabilized video**. When a video sequence is stabilized, there will be missing parts in the perimeter of each frame. Since the hole does not have to be bounded, the method can be applied here as well. Fig. 16 shows such an application.
5. **Creation of video textures**. The method is also capable of creating large video textures from small ones. In Fig. 17, a small video sequence (32 frames) was extended to a larger one, both spatially and temporally.

7 Summary and Conclusions

We have presented an objective function for the completion of missing data and an algorithm to optimize it. The objective treats the data uniformly in all dimensions and the algorithm was demonstrated on the completion of several challenging video sequences. In this realm, the objective proved to be very suitable as it only relies on very small, local parts of information. It successfully solved problems with hundreds of thousands of unknowns. The algorithm takes advantage of the sparsity of the database within the huge space of all image patches to derive an update step.

The method provides a principled approach with a clear objective function that extends those used in other works. We have shown here that it can be seen as a variant of EM and we have also shown its relation to BP.

There are several possible improvements to this method. First, we did not attempt to prune the database in any way, even though this is clearly the bottleneck with regard to running time. Second, we applied the measure to windows

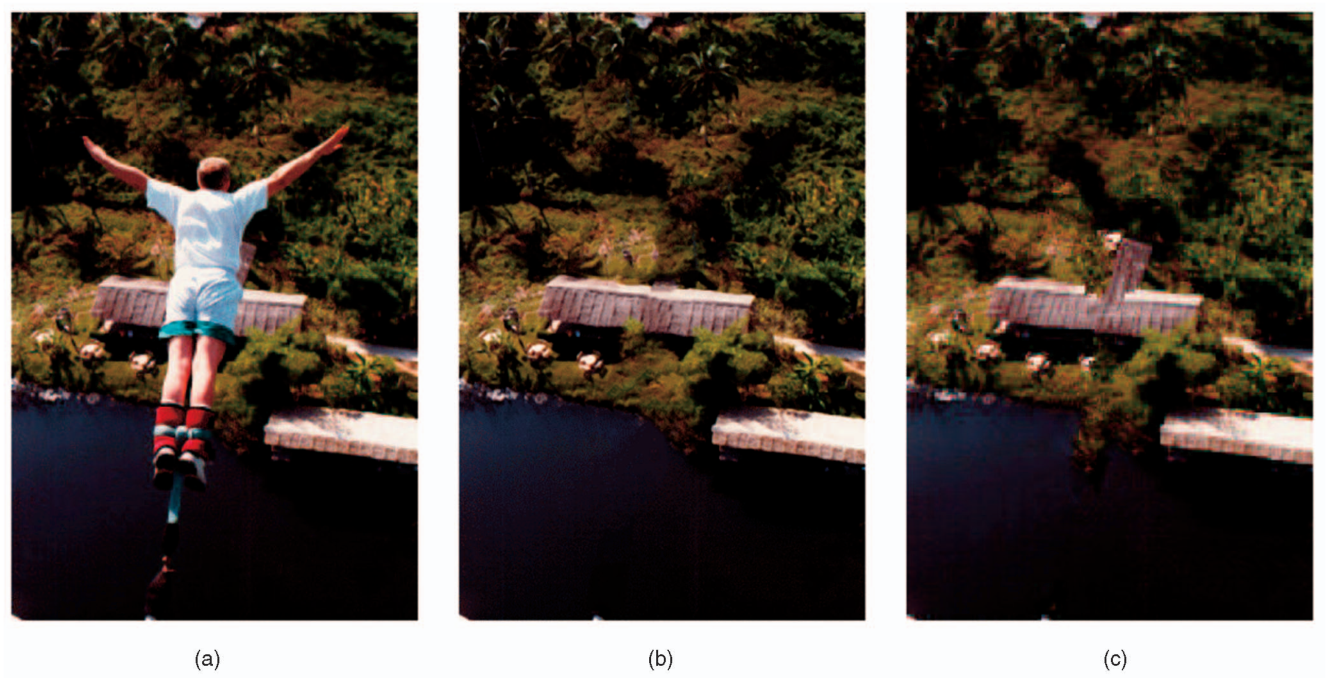

Fig. 12. Image completion example. (a) Original. (b) Our result. (c) Result from [9].

around every pixel, rather than a sparser set. Using a sparser grid (say, every third pixel) will give a significant speedup (3^3). Third, breaking the database into meaningful portions can prevent the algorithm from completing one class of data with another, even if they are similar (e.g., complete a person with background), as was done in [18], [24]. Fourth, the proposed method does not attempt to coalesce identical pixels that come from the same (static) part of the scene. These can be combined to form a joint constraint rather than being solved almost independently in each frame as was done here.

APPENDIX

In this section, we derive the optimization algorithm from Section 3 using the statistical estimation framework for the interested reader. In this framework, the global visual coherence of (1) can be derived as a likelihood function via a graphical model and our iterative optimization process can be viewed as a variant of the EM algorithm to find the maximum-likelihood estimate.

Under this model, the unknown parameters, denoted by Θ, are the color values of the missing pixel locations p in the space-time hole: $\Theta = \{c_p | p \in \mathcal{H}\}$. The known pixels around the hole are the observed variables $\mathbf{Y} = \{c_p | p \in \mathcal{S} \setminus \mathcal{H}\}$. The windows $\{W_n\}_1^N$ are defined as small space-time patches overlapping the hole where N is their total number. The set of patches $\mathbf{X} = \{X_n\}_1^N$ are the hidden variables corresponding to W_n. The space of possible assignments for each X_n are the data set patches in $\mathcal{D}$. We assume that each X_n can have any assignment with some probability. For the completion examples here, the data set is composed of all the patches from the same sequence that are completely known, i.e., $\mathbf{X} = \{X_n | X_n \subset \mathcal{S} \setminus \mathcal{H}\}$, but the data set may also contain patches from another source.

This setup can be described as a graphical model which is illustrated in Fig. 18 in one dimension. Each patch W_n is associated with one hidden variable X_n. This, in turn, is connected to all the pixel locations it covers, $p \in W_n$. As each W_n denotes a patch overlapping the hole, at least some of

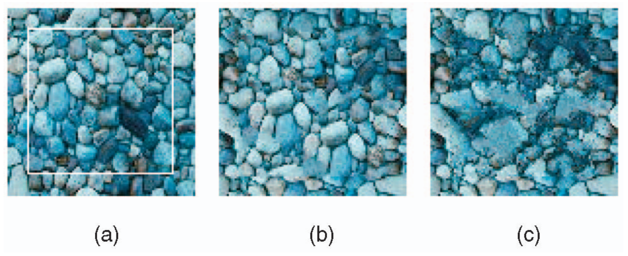

Fig. 13. Texture synthesis example. The original texture (a) was rotated 90 degrees and its central region (marked by white square) was filled using the original input image. (b) Our result. (c) Best results of (our implementation of) [14] were achieved with 9×9 windows (the only user defined parameter). Although the window size is large, the global structure is not maintained.

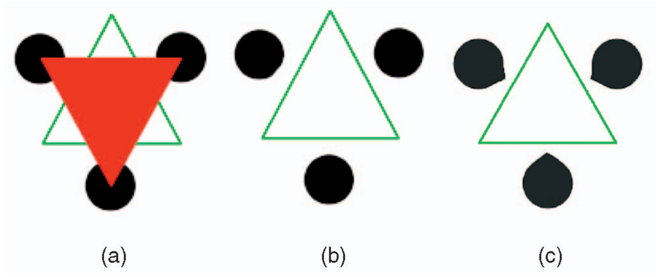

Fig. 14. The Kanitzsa triangle example is taken from [9] and compared with the algorithm here. The preference of [9] to complete straight lines creates the black corners in (c) which do not appear in (b). (a) Input. (b) Our result. (c) Result from [9].

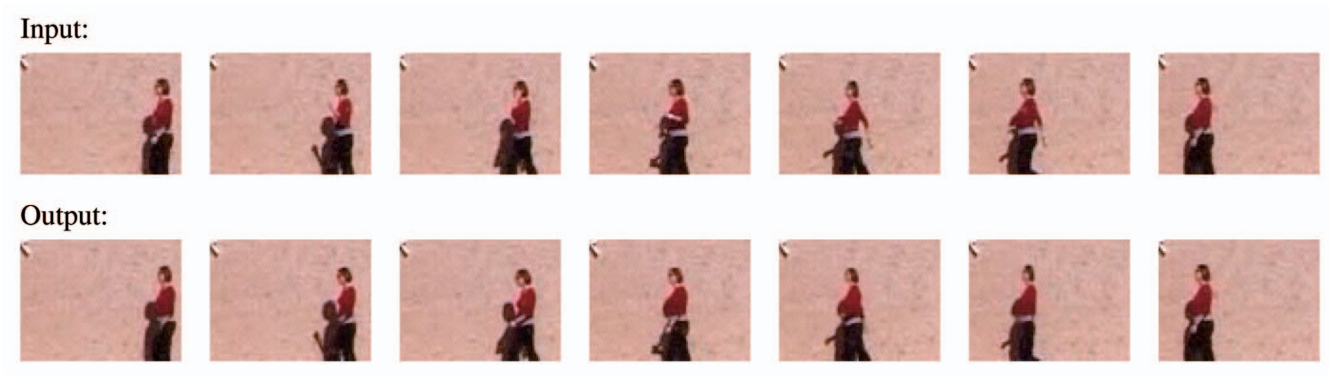

Fig. 15. Modification of a visual story. The woman in the top row scratches her ear but in the bottom row she does not. The video size is $90 \times 360 \times 190$ with 70,305 missing pixels.

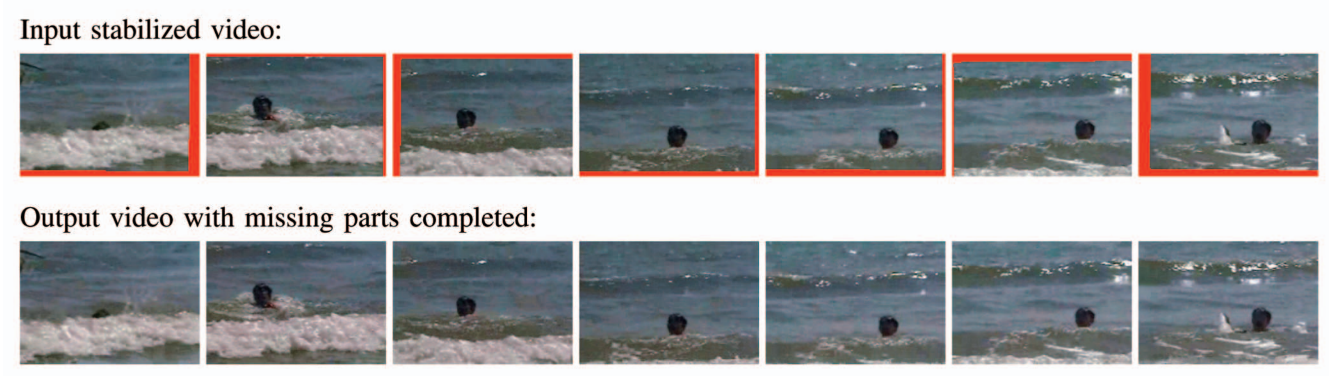

Fig. 16. Stabilization example. The top row shows a stabilized video sequence (courtesy of Matsushita [23]) of size $240 \times 352 \times 91$ with 466,501 missing pixels. Each frame was transformed to cancel the camera motion and, as a result, parts of each frame are missing (shown here in red). Traditionally, the video would be cropped to avoid these areas and to keep only the region that is visible throughout. Here, the missing parts are filled up using our method to create video with the full spatial extent. The results here are comparable with those in [23].

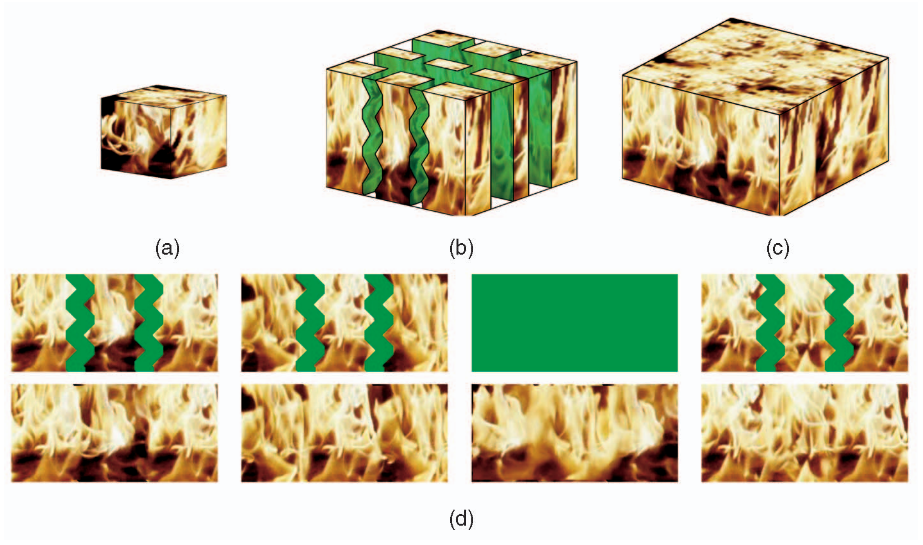

Fig. 17. Video texture synthesis. (a) A small video (sized $128 \times 128 \times 32$) is expanded to a larger one, both in space and in time. (b) is the larger volume of $128 \times 270 \times 60$ frames with 1,081,584 missing pixels. Small pieces of (a) were randomly placed in the volume with gaps between them. In (c), the gaps were filled automatically by the algorithm. Several frames from the initial and resulting videos are shown in (d). (a) Input video texture. (b) Initial volume. (c) Output video texture. (d) Initial and final resulting texture.

these pixel locations are unknowns, i.e., they belong to Θ. Let $c_p = W_n^p$ denote the color of the pixel p in the appropriate location within the window W_n and let X_n^p denote the color at the same location within the data set windows. Note that, while the color values X_n^p corresponding to pixel p may vary for different window locations n, the actual pixel color c_p is the *same* for all overlapping windows W_n^p. Using these notations, an edge in the graph that connects an unknown pixel p with an overlapping window X_n has an edge potential of the form $\phi(c_p, X_n) = e^{-\frac{1}{2\sigma^2}(c_p - X_n^p)^2}$.

The graph in Fig. 18 with the definition of its edge potentials is equivalent to the following joint probability

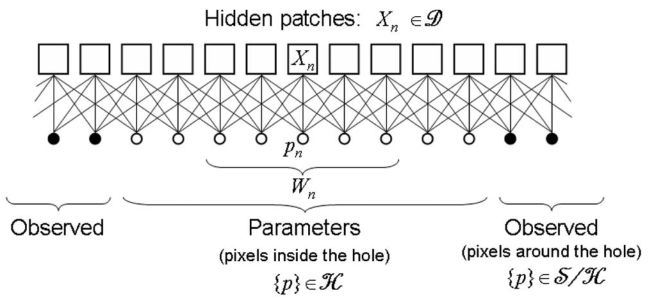

Fig. 18. Our graphical model for the completion problem.

density of the observed boundary variables and the hidden patches given the parameters Θ:

$$f(\mathbf{Y}, \mathbf{X}; \Theta) = \beta \prod_{n=1}^{N} \prod_{p \in W_n} \phi(c_p, X_n) = \beta \prod_{n=1}^{N} \prod_{p \in W_n} e^{-\frac{(c_p - X_n^p)^2}{2\sigma^2}},$$

where p denotes here either a missing or observed pixel and β is a constant normalizing the product to 1. This product is exactly the similarity measure defined previously in (2):

$$f(\mathbf{Y}, \mathbf{X}; \Theta) = \beta \prod_{n=1}^{N} e^{-d(X_n, W_n)} = \beta \prod_{n=1}^{N} \mathrm{sim}(X_n, W_n).$$

To obtain the likelihood function, we need to marginalize over the hidden variables. Thus, we need to integrate over all possible assignments for the hidden variables $X_1 \dots X_N$:

$$L = f(\mathbf{Y}; \Theta) = \sum_{(X_1, \dots, X_N) \in \mathcal{D}^N} f(\mathbf{Y}, \mathbf{X}; \Theta) \tag{5}$$

$$= \sum_{(X_1, \dots, X_N) \in \mathcal{D}^N} \beta \prod_{n=1}^{N} \mathrm{sim}(X_n, W_n). \tag{6}$$

The maximum-likelihood solution to the completion problem under the above model assumptions is the set of hole pixel values (Θ) that maximize this likelihood function. Note that, since the summation terms are products of all patch similarities, we will assume that this sum is dominated by the maximal product value (deviation of one of the patches from its best match will cause a significant decrease of the entire product term): $\max L \approx \max_{\{\mathbf{X}\}} \beta \prod_{n=1}^{N} \mathrm{sim}(X_n, W_n)$. Given the point values, seeking the best patch matches can be done independently for each W_n; hence, we can change the order of the max and product operators:

$$\max L \approx \beta \prod_{n=1}^{N} \max_{X_n} sim(X_n, W_n),$$

meaning that the *maximum-likelihood* solution is the same completion that attains best *visual coherence* according to (1).

We will next show that our optimization algorithm fits the EM algorithm [10] for maximizing the above likelihood function.

In the $\mathbf{E}$ step, at iteration t, the posterior probability density $\hat{\pi}^t$ is computed. Due to conditional independence of the hidden patches, this posterior can be written as the following product:

$$\hat{\pi}^t = f(\mathbf{X}|\mathbf{Y}; \Theta^t) = \prod_{n=1}^{N} f(X_n|\mathbf{Y}; \Theta^t) = \prod_{n=1}^{N} \beta_n \mathrm{sim}(X_n, W_n).$$

Thus, we get a probability value $\hat{\pi}_t$ for each possible assignment of the data set patches. Given the above considerations, these probabilities vanish in all but the best match assignments in each pixel, thus, the resulting PDF is an indicator function. This common assumption, also known as "Hard EM," justifies the choice of one nearest neighbor.

In the $\mathbf{M}$ step, the current set of parameters Θ are estimated by maximizing the following function:

$$\hat{\Theta}^{t+1} = \arg\max_{\Theta} \left(\sum_{(X_1, \dots, X_N) \in \mathcal{D}^N} \hat{\pi}^t \log f(\mathbf{X}, \mathbf{Y}; \Theta) \right).$$

Given the graphical model in Fig. 18, each unknown p depends only on its overlapping patches and the pixels are conditionally independent. Thus, the global maximization may be separated to local operations on each pixel:

$$\hat{c}_p^{t+1} = \arg\max_{c_p} \left(- \sum_{\{n \mid p \in W_n\}} (c_p - \hat{X}_n^p)^2 \right),$$

where the $\hat{X}_n$ are the best patch assignments for the current iteration (denoted also by V in Section 3.1). This is an L_2 distance between the point value and the corresponding color values in all covering patches and it is maximized by the *mean* of these values similar to (3).

Similar to the classic EM, the likelihood in the "Hard EM" presented here is increased in each $\mathbf{E}$ and $\mathbf{M}$ steps. Thus, the algorithm converges to some local maxima. The use of a multiscale process (see Section 3.2) and other adjustments (see Section 3.1) leads to a quick convergence and to a realistic solution with high likelihood.

ACKNOWLEDGMENTS

The authors would like to thank Oren Boiman for pointing out the graphical model that describes their algorithm and the anonymous reviewers for their helpful remarks. This work was supported in part by the Israeli Science Foundation (Grant No. 267/02) and by the Moross Laboratory at the Weizmann Institute of Science.

REFERENCES

[1] S. Arya and D.M. Mount, "Approximate Nearest Neighbor Queries in Fixed Dimensions," *Proc. Fourth Ann. ACM-SIAM Symp. Discrete Algorithms (SODA '93),* pp. 271-280, 1993.
[2] M. Ashikhmin, "Synthesizing Natural Textures," *Proc. 2001 Symp. Interactive 3D Graphics,* pp. 217-226, 2001.
[3] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman, "Texture Mixing and Texture Movie Synthesis Using Statistical Learning," *IEEE Trans. Visualization and Computer Graphics,* vol. 7, pp. 120-135, 2001.
[4] M. Bertalmío, A.L. Bertozzi, and G. Sapiro, "Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 355-362, Dec. 2001.
[5] M. Bertalmío, G. Sapiro, V. Caselles, and C. Ballester, "Image Inpainting," *ACM Proc. SIGGRAPH '00,* pp. 417-424, 2000.
[6] T. Chan, S.H. Kang, and J. Shen, "Euler's Elastica and Curvature Based Inpainting," *J. Applied Math.,* vol. 63, no. 2, pp. 564-592, 2002.

[7] V. Cheung, B.J. Frey, and N. Jojic, "Video Epitomes," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 42-49, 2005.
[8] D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach toward Feature Space Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 5, pp. 603-619, May 2002.
[9] A. Criminisi, P. Pérez, and K. Toyama, "Object Removal by Exemplar-Based Inpainting," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 721-728, June 2003.
[10] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc. B,* vol. 39, pp. 1-38, 1977.
[11] G. Doretto, A. Chiuso, Y. Wu, and S. Soatto, "Dynamic Textures," *Proc. IEEE Int'l Conf. Computer Vision,* vol. 51, no. 2, pp. 91-109, 2003.
[12] I. Drori, D. Cohen-Or, and H. Yeshurun, "Fragment-Based Image Completion," *ACM Trans. Graphics,* vol. 22, pp. 303-312, 2003.
[13] A. Efros and W.T. Freeman, "Image Quilting for Texture Synthesis and Transfer," *ACM Proc. SIGGRAPH '01,* pp. 341-346, 2001.
[14] A. Efros and T. Leung, "Texture Synthesis by Non-Parametric Sampling," *Int'l J. Computer Vision,* vol. 2, pp. 1033-1038, 1999.
[15] A.W. Fitzgibbon, Y. Wexler, and A. Zisserman, "Image-Based Rendering Using Image-Based Priors," *Int'l J. Computer Vision,* vol. 63, no. 2, pp. 141-151, 2005.
[16] W.T. Freeman, E.C. Pasztor, and O.T. Carmichael, "Learning Low-Level Vision," *Int'l J. Computer Vision,* vol. 40, no. 1, pp. 25-47, 2000.
[17] M. Irani and S. Peleg, "Motion Analysis for Image Enhancement: Resolution, Occlusion, and Transparency," *J. Visual Comm. and Image Representation,* vol. 4, pp. 324-335, 1993.
[18] J. Jia, T.P. Wu, Y.W. Tai, and C.K. Tang, "Video Repairing: Inference of Foreground and Background under Severe Occlusion," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 364-371, 2004.
[19] A. Kokaram, "Practical, Unified, Motion and Missing Data Treatment in Degraded Video," *J. Math. Imaging and Vision,* vol. 20, nos. 1-2, pp. 163-177, 2004.
[20] A.C. Kokaram, B. Collis, and S. Robinson, "Automated Rig Removal with Bayesian Motion Interpolation," *IEEE Proc.—Vision, Image, and Signal Processing,* vol. 152, no. 4, pp. 407-414, Aug. 2005.
[21] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut Textures: Image and Video Synthesis Using Graph Cuts," *ACM Trans. Graphics,* vol. 22, pp. 277-286, 2003.
[22] A. Levin, A. Zomet, and Y. Weiss, "Learning How to Inpaint from Global Image Statistics," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 305-312, 2003.
[23] Y. Matsushita, E. Ofek, X. Tang, and H.Y. Shum, "Full-Frame Video Stabilization," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 50-57, 2005.
[24] K.A. Patwardhan, G. Sapiro, and M. Bertalmio, "Video Inpainting of Occluding and Occluded Objects," *Proc. IEEE Int'l Conf. Image Processing,* pp. 69-72, Sept. 2005.
[25] A. Schödl and I. Essa, "Controlled Animation of Video Sprites," *Proc. First ACM Symp. Computer Animation,* (held in conjunction with ACM SIGGRAPH '02), pp. 121-127, 2002.
[26] A. Schödl, R. Szeliski, D.H. Salesin, and I. Essa, "Video Textures," *Proc. ACM SIGGRAPH '00,* pp. 489-498, 2000.
[27] U. Trottenber, C. Oosterlee, and A. Schüller, *Multigrid.* Academic Press, Inc., 2000.
[28] L. Wei and M. Levoy, "Fast Texture Synthesis Using Tree-Structured Vector Quantization," *Proc. ACM SIGGRAPH '00,* pp. 479-488, 2000.
[29] Y. Wexler, E. Shechtman, and M. Irani, "Space-Time Video Completion," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 120-127, 2004.

Yonatan Wexler received the BSc degree in mathematics and computer science from the Hebrew University in 1996 and the PhD degree from the University of Maryland in 2001. From 2001 to 2003, he was a member of Oxford's Visual Geometry Group and, from 2003 to 2005, he was a researcher at the Weizmann Institute of Science. He received the Marr Prize best paper award at ICCV '03 and a best poster award at CVPR '04. He is a member of the IEEE Computer Society.

Eli Shechtman received the BSc degree magna cum laude in electrical engineering from Tel-Aviv University in 1996, and the MSc degree in mathematics and computer science from the Weizmann Institute of Science in 2003. He received the Weizmann Institute Dean's Prize for MSc students, the best paper award at the European Conference on Computer Vision (ECCV '02), and a best poster award at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '04). He is currently a PhD student at the Weizmann Institute of Science. His current research focuses on video analysis and its applications. He is a student member of the IEEE Computer Society.

Michal Irani received the BSc degree in mathematics and computer science in 1985 and the MSc and PhD degrees in computer science in 1989 and 1994, respectively, all from the Hebrew University of Jerusalem. From 1993 to 1996, she was a member of the technical staff in the Vision Technologies Laboratory at the David Sarnoff Research Center, Princeton, New Jersey. She is currently an associate professor in the Department of Computer Science and Applied Mathematics at the Weizmann Institute of Science, Israel. Her research interests are in the areas of computer vision and video information analysis. Dr. Irani's prizes and honors include the David Sarnoff Research Center Technical Achievement Award (1994), the Yigal Allon three-year fellowship for outstanding young scientists (1998), and the Morris L. Levinson Prize in Mathematics (2003). She also received the best paper awards at ECCV '00 and ECCV '02 (European Conference on Computer Vision), the honorable mention for the Marr Prize at ICCV '01 and ICCV '05 (IEEE International Conference on Computer Vision), and a best poster award at CVPR '04 (Computer Vision and Pattern Recognition). She served as an associate editor of *TPAMI* from 1999-2003. She is a member of the IEEE Computer Society.