

Motion Denoising with Application to Time-lapse Photography

Michael Rubinstein¹ Ce Liu² Peter Sand Frédo Durand¹ William T. Freeman¹
¹MIT CSAIL ²Microsoft Research New England

{mrub, sand, fredo, billf}@mit.edu celiu@microsoft.com

Abstract

Motions can occur over both short and long time scales. We introduce motion denoising, which treats short-term changes as noise, long-term changes as signal, and re-renders a video to reveal the underlying long-term events. We demonstrate motion denoising for time-lapse videos. One of the characteristics of traditional time-lapse imagery is stylized jerkiness, where short-term changes in the scene appear as small and annoying jitters in the video, often obfuscating the underlying temporal events of interest. We apply motion denoising for resynthesizing time-lapse videos showing the long-term evolution of a scene with jerky short-term changes removed. We show that existing filtering approaches are often incapable of achieving this task, and present a novel computational approach to denoise motion without explicit motion analysis. We demonstrate promising experimental results on a set of challenging time-lapse sequences.

1. Introduction

Short-term, random motions can distract from the longer-term changes recorded in a video. This is especially true for time-lapse sequences where dynamic scenes are captured over long periods of time. When day- or even year-long events are condensed into minutes or seconds, the pixels can be temporally inconsistent due to the significant time aliasing. Such aliasing effects take the form of objects suddenly appearing and disappearing, or illumination changing rapidly between consecutive frames.

It is therefore important to remove random, temporally inconsistent motion. We want to design a video processing system that removes temporal jitters and inconsistencies in an input video, and generates a temporally smooth video as if randomness never appeared. We call this technique *motion denoising* in analogy to image denoising. Since visual events are decomposed into slow-varying and fast-changing components in motion denoising, such technique can be useful for time-lapse photography, which is widely used in the movie industry especially for documentary movies, but

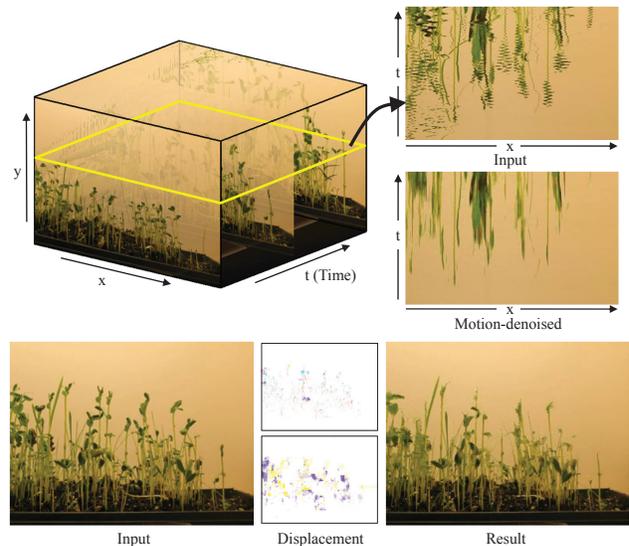


Figure 1. A time-lapse video of plants growing (sprouts). XT slices of the video volumes are shown for the input sequence and for the result of our motion denoising algorithm (top right). The motion-denoised sequence is generated by spatiotemporal rearrangement of the pixels in the input sequence (bottom center; spatial and temporal displacement on top and bottom respectively, following the color coding in Figure 5). Our algorithm solves for a displacement field that maintains the long-term events in the video, while removing the short-term noisy motions. The full sequence and results are available in the accompanying material.

has also become prevalent among personal users. It can also assist long-period medical and scientific analysis. For the rest of the paper we will refer to motion denoising, and its induced motion decomposition, interchangeably.

Motion denoising is by no means a trivial problem. Previous work on motion editing has been focusing on accurately measuring the underlying motion and carefully constructing coherent motion layers in the scene. These kind of motion-based techniques are often not suitable for analyzing time-lapse videos. The jerky nature of these sequences violates the core assumptions of motion analysis and optical flow, and prevent even the most sophisticated motion estimation algorithm from obtaining accurate enough motion for further analysis.

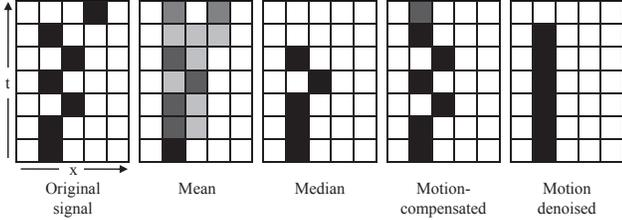


Figure 2. The responses of different filters on a canonical signal with a temporal dimension and a single spatial dimension. In this example, the temporal filters are of size 3, centered at the pixel, and the denoising algorithm uses a 3×3 support. For the motion compensated filter, we assume the temporal trajectory is accurate until $t = 6$. That is, the motion of the pixel from $(x, t) = (2, 6)$ to $(4, 7)$ was not detected correctly.

We propose a novel computational approach to motion denoising in videos that does not require explicit motion estimation and modeling. We formulate the problem in a Bayesian framework, where the goal is to recover a “smooth version” of the input video by reshuffling its pixels spatiotemporally. This translates to a well-defined inference problem over a 3D Markov Random Field (MRF), which we solve using a time-space optimized Loopy Belief Propagation (LBP) algorithm. We show how motion denoising can be used to eliminate short-term motion jitters in time-lapse videos, and present results for time-lapse sequences of different nature and varying scenes.

2. Background and Related Work

The input to our system is an $M \times N \times T$ video sequence $I(x, y, t)$, with RGB intensities given in range $[0, 255]$. Our goal is to produce an $M \times N \times T$ output sequence $J(x, y, t)$ in which short term jittery motions are removed, whereas long term scene changes are maintained. A number of attempts have been made to tackle similar problems from a variety of perspectives.

Temporal filtering. A straightforward approach is to pass the sequence through a temporal low-pass filter

$$J(x, y, t) = f(I(x, y, \{k\}_{t-\delta_t}^{t+\delta_t})) \quad (1)$$

where f denotes the filtering operator, and δ_t defines the temporal window size. For a video sequence with static viewpoint, f is often taken as the median operator, which is useful for tasks such as background-foreground segmentation and noise reduction. This approach, albeit simple and fast, has an obvious limitation – the filtering is performed independently at each pixel. In a dynamic scene with rapid motion, pixels belonging to different objects are averaged, resulting in a blurred or discontinuous result. Figure 2 demonstrates this on a canonical signal. Figure 3 further illustrates these effects on a natural time-lapse video.

To address this issue, motion-compensated filtering was introduced, filtering the sequence along motion trajectories

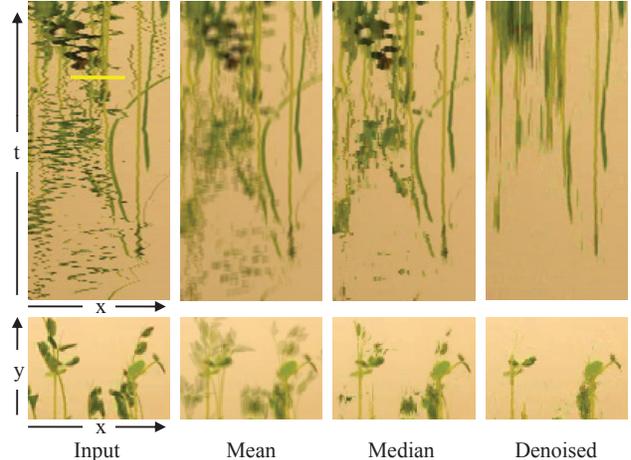


Figure 3. Comparison with temporal filtering of the plants sequence of Figure 1. Zoom-in on the left part of the XT slice from Figure 1 are shown for the original sequence, the (temporally) mean-filtered sequence, the median-filtered sequence, and the motion-denoised sequence. At the bottom, a representative spatial patch from each sequence is shown, taken from a region roughly marked by the yellow bar on the input video slice.

(e.g. [13]). Such techniques are commonly employed for video compression, predictive coding and noise removal. Although this approach is able to deal with some of the blur and discontinuity artifacts as pixels are only integrated along the estimated motion path, it does not filter the actual motion (i.e. making the motion trajectory smoother), but rather takes the motion into account for filtering the sequence. In addition, errors in motion estimation (e.g. optical flow) may result in unwanted artifacts at object boundaries.

Motion editing. One work that took a direct approach to motion editing in videos is “Motion Magnification” [8]. A layer segmentation system was proposed for exaggerating motions that might otherwise be difficult or even impossible to notice. Similar to our work, they also manipulate video data to resynthesize a sequence with modified motions. However, our work modifies the motion without explicit motion analysis or layer modeling that are required by their method. In fact, layer estimation can be challenging for the sequences we are interested in; these sequences may contain too many layers (e.g. leaves) for which even state-of-the-art layer segmentation algorithms would have difficulties producing reliable estimates.

Motion denoising has been addressed before in a *global* manner, known as video stabilization (e.g. [11, 10]). Camera jitter (mostly from hand-held devices) is modeled using image-level transforms between consecutive frames, which are used to synthesize a sequence with smoother camera motion. Image and video inpainting techniques are often utilized to fill-in missing content in the stabilized sequence. In contrast, we focus on stabilization at the *object level*, sup-

porting pixel-wise displacement within a single frame. Inpainting is built-in naturally into our formulation.

Time-lapse videos. Previous work on time-lapse videos use time-lapse sequences as an efficient representation for video summarization. Work such as [3, 15] take a video-rate footage as input, and output a sequence of frames that succinctly depict temporal events in the video. Work such as [15, 16] can take an input time-lapse video, and produce a single-frame representation of that sequence, utilizing information from throughout the time span. Time-lapse sequences have also been used for other applications. For example, Weiss [22] uses a sequence of images of a scene under varying illumination to estimate intrinsic images. These techniques and applications are significantly different from ours. Both input and output of our system are time-lapse videos, and our goal is to improve the input sequence quality by suppressing short-term distracting events.

Direct editing of time-lapse imagery has been proposed in [18] by factorizing each pixel in an input time-lapse video into shadow, illumination and reflectance components, which can be used for relighting or editing the sequence, and for recovering the scene geometry. In this paper, we are interested in a different type of decomposition: separating a time-lapse sequence into shorter- and longer-term events. These two types of decompositions can be complimentary to each other.

Geometric rearrangement. In the core of our method is a statistical (data-driven) algorithm for inferring smooth motion from noisy motion by rearranging the input video. Content rearrangement in images and video has been used in the past for various editing applications. Image reshuffling is discussed in [5]. They work in patch space, and the patch moves are constrained to an underlying coarse grid which does not allow handling relatively small motions common to time-lapse videos. We, on the other hand, work in pixel resolution, supporting both small and irregular displacements. [14] perform geometric image rearrangement for various image editing tasks using a MRF formulation. Our work concentrates on a different problem and requires different formulation. Inference in videos is much more challenging than in images, and we use different inference tools from the ones used in their work.

For videos, [17] consider the sequence appearance and dynamics for shifting entire frames to create a modified playback. [15] generate short summaries for browsing and indexing surveillance data by shifting individual pixels temporally while keeping their spatial locations intact. [21] align two videos using affine spatial and temporal warps. Our problem is again very different from all these work. Temporal shifts alone are insufficient to denoise noisy motion, and spatial offsets must be defined in higher granularity than the global frame or video. This makes the problem much more challenging to solve.

3. Formulation

We model the world as evolving slowly through time. That is, within any relatively small time span, objects in the scene attain some stable configuration, and changes to these configurations occur in low time rates. Moreover, we wish to make use of the large redundancy in images and videos to infer those stable configurations. This leads to the following formulation. Given an input video I , we seek an output video J that minimizes the energy $E(J)$, defined as

$$E(J) = \sum_{x,y,t} |J(x, y, t) - I(x, y, t)| + \alpha \sum_{x,y,t} |J(x, y, t) - J(x, y, t + 1)| \quad (2)$$

subject to

$$J(x, y, t) = I(x + w_x(x, y, t), y + w_y(x, y, t), t + w_t(x, y, t)) \quad (3)$$

for spatiotemporal displacement field

$$w(x, y, t) \in \{(\delta_x, \delta_y, \delta_t) : |\delta_x| \leq \Delta_s, |\delta_y| \leq \Delta_s, |\delta_t| \leq \Delta_t\} \quad (4)$$

where (Δ_s, Δ_t) are parameters defining the support (search) region.

In this objective function, the first term is a *fidelity term*, enforcing the output sequence to resemble the input sequence at each location and time. The second term is a *temporal coherence* term, which requires the solution to be temporally smooth. The tension between those two terms creates a solution which maintains the general appearance of the input sequence, and is temporally smooth. This tradeoff between appearance and temporal coherence is controlled via the parameter α .

As J is uniquely defined by the spatiotemporal displacements, we can equivalently rewrite Equation 2 as an optimization on the displacement field w . Further parameterizing $p = (x, y, t)$, and plugging constraint 3 into Equation 2, we get

$$E(w) = \sum_p |I(p + w(p)) - I(p)| + \alpha \sum_{p,r \in \mathcal{N}_t(p)} ||I(p + w(p)) - I(r + w(r))||^2 + \gamma \sum_{p,q \in \mathcal{N}(p)} \lambda_{pq} |w(p) - w(q)| \quad (5)$$

where we also added an additional term for regularizing the displacement field w , with weight $\lambda_{pq} = \exp\{-\beta ||I(p) - I(q)||^2\}$. β is learnt as described in [19]. λ_{pq} assigns varying weight to discontinuities in the displacement map, as function of the similarity between neighboring pixels in the original video. $\mathcal{N}(p)$ denotes the spatiotemporal neighborhood of pixel p , and $\mathcal{N}_t(p) \subseteq \mathcal{N}(p)$ denotes the temporal

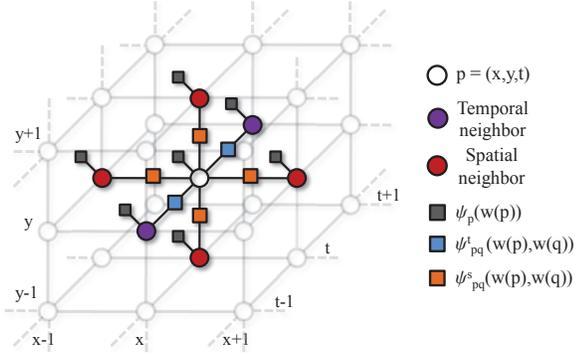


Figure 4. An illustration of the graphical model corresponding to Equation 5. Note that each node contains the three (unknown) components of the spatiotemporal displacement at that location.

neighborhood of p . We use the six spatiotemporal pixels directly connected to p as the neighborhood system.

α and γ weight the temporal coherence and regularization terms respectively. The L_2 norm is used for temporal coherence to discourage motion discontinuities, while L_1 is used in the fidelity and regularization terms to remove noise from the input sequence, and to account for discontinuities in the displacement field, respectively.

3.1. Optimization

We optimize Equation 5 discretely on a 3D MRF corresponding to the three-dimensional video volume, where each node p corresponds to a pixel in the video sequence, and represents the latent variables $w(p)$. The state space in our model is the set of possible three-dimensional displacements within a predefined search region (Equation 4). The potential functions are given by

$$\psi_p(w(p)) = |I(p + w(p)) - I(p)| \quad (6)$$

$$\psi_{pr}^t(w(p), w(r)) = \alpha \|I(p + w(p)) - I(r + w(r))\|^2 + \gamma \lambda_{pr} |w(p) - w(r)| \quad (7)$$

$$\psi_{pq}^s(w(p), w(q)) = \gamma \lambda_{pq} |w(p) - w(q)| \quad (8)$$

where ψ_p is the unary potential at each node, and ψ_{pr}^t, ψ_{pq}^s denote the temporal and spatial pairwise potentials respectively. Figure 4 depicts the structure of this graphical model.

We have experimented with several optimization techniques for solving Equation 5, namely Iterated Conditional Modes (ICM), α -expansion (GCUT) [4] and loopy belief propagation (LBP) [23]. Our temporal pairwise potentials (Equation 7) are neither a metric nor a semi-metric, which makes the graph-cut based algorithms theoretically inapplicable to this optimization. Although previous work employ those algorithms ignoring the metric constraints and still report good results (*e.g.* [14]), our experiments consistently showed that LBP manages to produce more visually appealing sequences, and in most cases also achieves lower energy

solutions comparing to the other solvers. We choose LBP as our inference engine.

Figure 5 compares the results of the three optimizations. The complete sequences are available in the supplementary material. The ICM results suffer, as expected, from noticeable discontinuities. There are also noticeable artifacts in the graph-cut solution. As part of our pairwise potentials are highly non metric, it is probable that α -expansion will make incorrect moves, which adversely affect the results. All three methods tend to converge quickly within 3 – 5 iterations, which agrees with the related literature [19]. Although the solution energies tend to be within the same ballpark, we noticed they usually correspond to different local minima.

3.2. Implementation Details

Our underlying graphical model is a massive 3D grid, which imposes computational difficulties in both time and space. For tractable runtime, we trivially extend the message update schedule by Tappen and Freeman [20] to 3D, by first sending messages (forth and back) along rows in all frames, then along columns, and finally along time. This sequential schedule allows information to propagate quickly through the grid, and helps the algorithm converge faster. To search larger ranges, we apply LBP to a spatiotemporal video pyramid. Since time-lapse sequences are temporally aliased, we apply smoothing to the spatial domain only, and sample in the temporal domain. At the coarser level, the same search volume effectively covers twice the volume used in the finer level, allowing the algorithm to consider larger spatial and temporal ranges. To propagate the displacements to the finer level, we bilinear-interpolate and scale (multiply by 2) the shifts, and use them as centers of the search volume at each pixel in the finer level.

The complexity of LBP is linear in the graph size, but quadratic in the state space. In our model, we have a K^3 search volume (for $\Delta_s = \Delta_t = K$), which requires K^6 computations per message update and may quickly become intractable even for relatively small search volumes. Nevertheless, we can get significant speedup in the computation of the spatial messages using distance transform, as the 3D displacement components are decoupled in the L_1 -norm distance. Felzenszwalb and Huttenlocher have shown that computing a distance transform on such 3D label grids can be reduced to consecutive efficient computations of 1D distance transforms [6]. The overall complexity of this computation is $O(3K^3)$, which is linear in the search range. For our multiscale computation, Liu *et al.* [9] already showed how the distance transform can be extended to handle offsets. Following the reduction in [6] therefore shows that we can trivially handle offsets in the 3D case as well. We note that the distance transform computation is not an approximation, and results in the exact message updates.

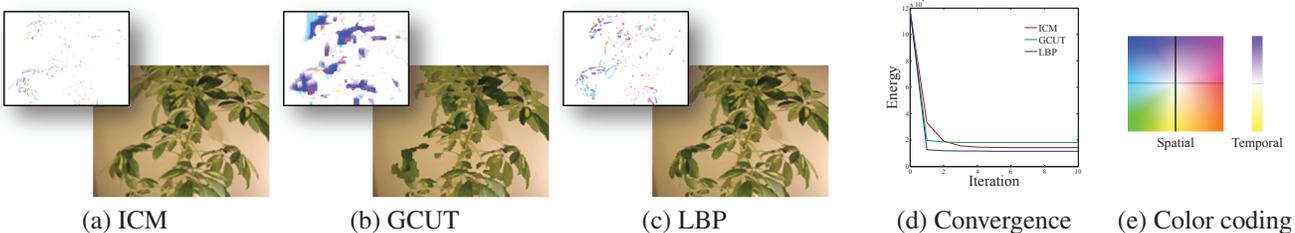


Figure 5. Comparison between different optimization techniques for solving Equation 5, demonstrated on the `plant` sequence (Figure 8). (a-c) Representative frame from each result. The spatial components of the displacement fields (overlaid) illustrate that different local minima are attained (the full sequences are available in the accompanying material). (d) The energy convergence pattern over 10 iterations of the algorithms. (e) The color coding used for visualizing the displacement fields, borrowed from [2]. To avoid introducing an additional color coding for the temporal displacements, the colors along the y -axis are used to represent backward ($-y$) to forward ($+y$) temporal displacement.

This yields significant improvement in running time as 2/3 of the messages propagating through the graph can be computed in linear time. Unfortunately, we cannot follow a similar approach for the temporal messages, as the temporal pairwise potential function (Equation 7) is non-convex.

This massive inference problem imposes computational difficulties in terms of space as well. For example, a 500^3 video sequence with a 10^3 search region requires memory, for the messages only, of size at least $500^3 \times 10^3 \times 6 \times 4 \simeq 3$ terabytes (!). Far beyond current available RAM, and probably beyond the average available disk space. We therefore restrict our attention to smaller sequences and search volumes. As the messages structure cannot fit entirely in memory, we store it on disk, and read and write the necessary message chunks on need. For our message update schedule, it suffices to maintain in memory the complete message structure for one frame for passing messages spatially, and two frames for passing messages temporally. This imposes no memory difficulty even for larger sequences, but comes at the cost of lower performance as disk I/O is far more expensive than memory access. Section 4 details the algorithm’s space and time requirements for the videos and parameters we used.

Finally, once LBP converges or message passing is complete, the MAP label assignment is traditionally computed independently at each node [7]

$$\hat{w}_p = \arg \min_{w_p} \left(\psi_p(w_p) + \sum_{q \in \mathcal{N}(p)} m_{q \rightarrow p}(w_p) \right) \quad (9)$$

For our problem, we observe that the local conditional densities are often multi-modal (Figure 6), indicating multiple possible solutions that are equivalent, or close to equivalent, with respect to the objective function. The label (displacement) assigned to each pixel therefore depends on the order of traversing the labels, which is somewhat arbitrary. As a result, the selected displacements at neighboring pixels need not be coherent with respect to the pairwise potentials.

We propose a different procedure for assigning the MAP label at each node. Following our message update schedule,

we start with the pixel at position $(0, 0, 0)$ and assign it the label satisfying Equation 9. Then, traversing the grid from back to front, top to bottom and left to right, we assign each node the label according to

$$\hat{w}_p^* = \arg \min_{w_p} \left(\psi_p(w_p) + \sum_{q \in \mathcal{P}(p)} \psi_{pq}(w_p, \hat{w}_q^*) + \sum_{q \in \mathcal{N}(p) \setminus \mathcal{P}(p)} m_{q \rightarrow p}(w_p) \right) \quad (10)$$

where $\mathcal{P}(p)$ denotes the left, top, and backward neighbors of node p . Notice that these nodes were already assigned labels by the time node p is reached in this scan pattern, and so their assignments are used while determining the assignment for p . Overall, this process produces a MAP assignment that is locally more coherent with respect to the objective function. In practice, we found that the solutions produced with this approach have energies 2 – 3% lower on average comparing to independent assignment of states to pixels, and the decrease in energy is obviously larger when heavier regularization is sought.

4. Results

Our main application of interest is time-lapse processing, and so we ran the motion denoising algorithm on several time-lapse sequences of different nature. We fixed the parameters in all the experiments to $\alpha = 2$, $\gamma = 10$, search volume of size $7 \times 7 \times 5$, and a 2-level video pyramid. We ran LBP for 5 – 10 iterations, during which the algorithm converged to a stable minima. Representative frames for each experiment are shown in Figure 8, and the full sequences and results are available in the accompanying material.

Recall that our basic assumption is that the input sequence contains events of different time scales. We first produced sequences which demonstrate this effect in a controlled environment. First, we set up a Canon PowerShot series camera shooting a plant indoors. We used a fan for simulating wind, and a slow-moving light source for emulating a low-variation change in the scene. We then sampled

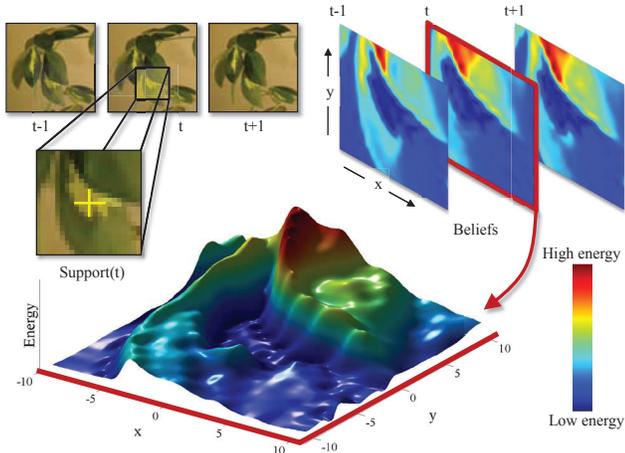


Figure 6. A visualization of the beliefs computed by LBP for a single pixel in the `plant` sequence, using a $31 \times 31 \times 3$ support. The support frame containing the pixel is zoomed-in on the upper left. The beliefs over the support are shown on the upper right, colored from blue (low energy) to red (high energy). We seek the displacement that has the minimal energy within the support. At the bottom, the belief surface is shown for the middle frame of the support region, clearly showing multiple equivalent (or nearly equivalent) solutions.

the captured video at a low frame rate to introduce aliasing in time. A sampling rate of 2 frames/sec allowed sufficient time for the leaves to move and create the typical motion jitter effect. As can be seen in the result (`plant`), our algorithm manages to find a stable and faithful configuration for the plant, while perfectly maintaining the lighting change in the scene.

`sprouts` illustrates the process of plants growing in a similar indoor setup. In this experiment, we set the camera to capture a still image every 15 minutes, and the sprouts were placed in an enclosure so that motions are created solely by the plants. The motion-denoised result appears smoother than the original sequence, and captures the growth process of the plants with the short-term noisy motions removed. It can be seen that for some parts of the plants the motions are stabilized (as also shown in the temporal slices of the video (Figures 1,8 right column), while other parts, namely the sprouts’ tops, are partly removed in the result. The motion at the top of the stems is the largest, and the algorithm is sometimes unable to infer the underlying stationary configuration using the search volumes we used. In such cases, the algorithm gracefully removes these objects and fills-in their place with the background or other objects they occlude, so as to generate a smooth looking sequence. Enlarging the support region will allow the algorithm to denoise larger motions (Figure 7), but has a large impact on its performance as will be discussed shortly.

Time-lapse videos span a large variety of scenes and styles. To test our technique on a different type of time-lapse sequences, we experimented with a sequence from the

Extreme Ice Survey (EIS) [1]. EIS documents the changes to the Earth’s glacial ice via time-lapse photography. We received their raw time-lapse capture of the Sólheimajökull glacier in Iceland, taken between April 2007 and May 2010 at a rate of 1 frame per hour (video available in the supplemental material). The characteristics of this sequence, and motions within it, are quite different than the ones we addressed before. Their end result, produced manually by a video editor, is quite cluttered due to changes in lighting and weather conditions in this extreme environment, which are difficult to handle manually.

To reduce the sequence size, and prevent the large scene variations from contaminating the inference, we first sampled this sequence using a non-uniform sampling technique similar to [3]. We found the gist descriptor [12] useful as a frame distance measure for computing the sampling, producing a more fluent sampling than working in color space directly using L_1 or L_2 distance. Our result on this sequence resembles the response of a temporal filter. Without apparent motions jitters, the best solution is to smooth the sequence temporally. Indeed, the computed displacements are dominated by the temporal component (Figure 8, bottom row), meaning that the algorithm reduces to temporal rearrangement. This, however, happens naturally without user intervention or parameter tuning.

More results are shown in Figure 8. `pool` and `pond` nicely illustrate how the algorithm manages to maintain temporal events in the scene while eliminating short term distracting motions. Notice how, in the `pool` sequence, the shadows are maintained on the porch and near the swing chair, and how the rapid motions of the tree in the back and flowers in the front are stabilized. Worthy of note is that the input sequences were downloaded from the web in moderate resolution, showing robustness of the algorithm to video quality and compression artifacts.

In `pond`, the algorithm manages to denoise the jittery motions in the water and vegetation, while maintaining the motion of the sun and its reflection on the water. Some artifacts are apparent, however, in the sky region. This is because the clouds’ dynamics does not fully fit our motion decomposition model – they neither exhibit jittery motion per se, nor evolve slowly as other objects (*e.g.* the sun) in the scene. Adding more careful treatment of such intermediate motion scales, or allowing the user to specify her desired motion scale of interest, are interesting directions for future work.

Despite our optimizations, the running time of the algorithm is still substantial. Solving for a sequence of size 300^3 with a 7^3 search volume takes approximately 50 hours for our CPU-based C++ implementation of LBP using sequential message passing, on a quad-core Intel Xeon 2.66GHZ desktop with 28GB RAM. For such dimensions, the algorithm makes use of 1GB of RAM, and up to 50GB scratch

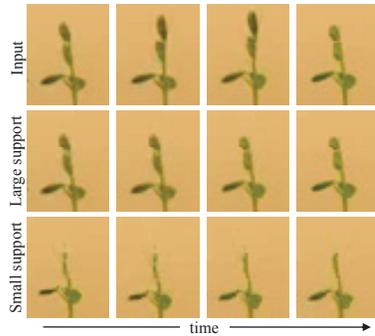


Figure 7. Zoom-in on the rightmost plant in the `sprouts` sequence in four consecutive frames shows that enlarging the search volume used by the algorithm can greatly improve the results. “Large support” corresponds to a $31 \times 31 \times 5$ search volume, while “small support” is the $7 \times 7 \times 5$ volume we used in our experiments.

space on disk. We used an Intel X-25M solid state drive, which gave an approximate x2 speedup in disk I/O, but the running time was dominated by the computation of the temporal messages. As this computation is quadratic in the size of the search volume, using larger volumes means even larger increase in the run time. We therefore resorted to using relatively small search volumes in our experiments.

Even with multi-scale processing, some objects might exhibit motions larger than the range scanned by the algorithm. In such cases they will be gracefully removed from the sequence, which might not necessarily be the desired result. Figure 7 demonstrates that this limitation is computational, rather than theoretical.

5. Conclusion

We have introduced motion denoising – the process of decomposing videos into long- and short-term motions, allowing motion resynthesis in a way that maintains one and removes the other. We showed how motion denoising can be cast as an inference problem within a well-defined data-driven formulation that does not require explicit motion estimation. This allows the algorithm to operate on videos containing highly involved dynamics. We also showed that Loopy Belief Propagation is a suitable machinery for performing this inference. Time-lapse videos fit particularly well within this decomposition model, and we presented a novel application whose goal is, given an input time-lapse sequence, to synthesize a new one in which the typical short term jittery motions are removed whilst the underlying long term evolution in the sequence is maintained. We presented results on a set of challenging time-lapse videos. Our technique successfully generates filtered timelapse sequences that are visually faithful to the long-term trends and allow a better grasp of the underlying long-term temporal events in the scene.

Acknowledgments

We thank Extreme Ice Survey for their glacier time-lapse video. This research is partially funded by NGA NEGI-1582-04-0004, Shell Research, ONR-MURI Grant N00014-06-1-0734, NSF 0964004, Quanta, and by gift from Microsoft, Adobe, and Google.

References

- [1] Extreme Ice Survey. <http://www.extremeicesurvey.org/>. 6
- [2] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *ICCV*, pages 1–8, 2007. 5
- [3] E. Bennett and L. McMillan. Computational time-lapse video. In *ACM SIGGRAPH*, volume 26, 2007. 3, 6
- [4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *TPAMI*, 23(11):1222–1239, 2002. 4
- [5] T. S. Cho, M. Butman, S. Avidan, and W. T. Freeman. The patch transform and its applications to image editing. In *CVPR*, 2008. 3
- [6] P. Felzenszwalb and D. Huttenlocher. Distance transforms of sampled functions. *Cornell Computing and Information Science Technical Report TR2004-1963*, 2004. 4
- [7] P. Felzenszwalb and D. Huttenlocher. Efficient belief propagation for early vision. *IJCV*, 70(1):41–54, 2006. 5
- [8] C. Liu, A. Torralba, W. Freeman, F. Durand, and E. Adelson. Motion magnification. In *ACM SIGGRAPH*, pages 519–526. ACM, 2005. 2
- [9] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing: Label transfer via dense scene alignment. In *CVPR*, pages 1972–1979, 2009. 4
- [10] F. Liu, M. Gleicher, H. Jin, and A. Agarwala. Content-preserving warps for 3D video stabilization. In *ACM SIGGRAPH*, pages 1–9, 2009. 2
- [11] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum. Full-frame video stabilization with motion inpainting. *TPAMI*, 28:1150–1163, 2006. 2
- [12] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001. 6
- [13] M. Ozkan, M. Sezan, and A. Tekalp. Adaptive motion-compensated filtering of noisy image sequences. volume 3, pages 277–290. IEEE, 2002. 2
- [14] Y. Pritch, E. Kav-Venaki, and S. Peleg. Shift-map image editing. In *ICCV*, pages 151–158, 2009. 3, 4
- [15] Y. Pritch, A. Rav-Acha, and S. Peleg. Nonchronological video synopsis and indexing. *TPAMI*, 30(11):1971–1984, 2008. 3
- [16] R. Raskar, A. Ilie, and J. Yu. Image fusion for context enhancement and video surrealism. In *ACM SIGGRAPH 2005 Courses*, page 4. 3
- [17] A. Schödl, R. Szeliski, D. Salesin, and I. Essa. Video textures. In *ACM SIGGRAPH*, pages 489–498, 2000. 3
- [18] K. Sunkavalli, W. Matusik, H. Pfister, and S. Rusinkiewicz. Factored Time-Lapse Video. In *ACM SIGGRAPH*, volume 26, 2007. 3

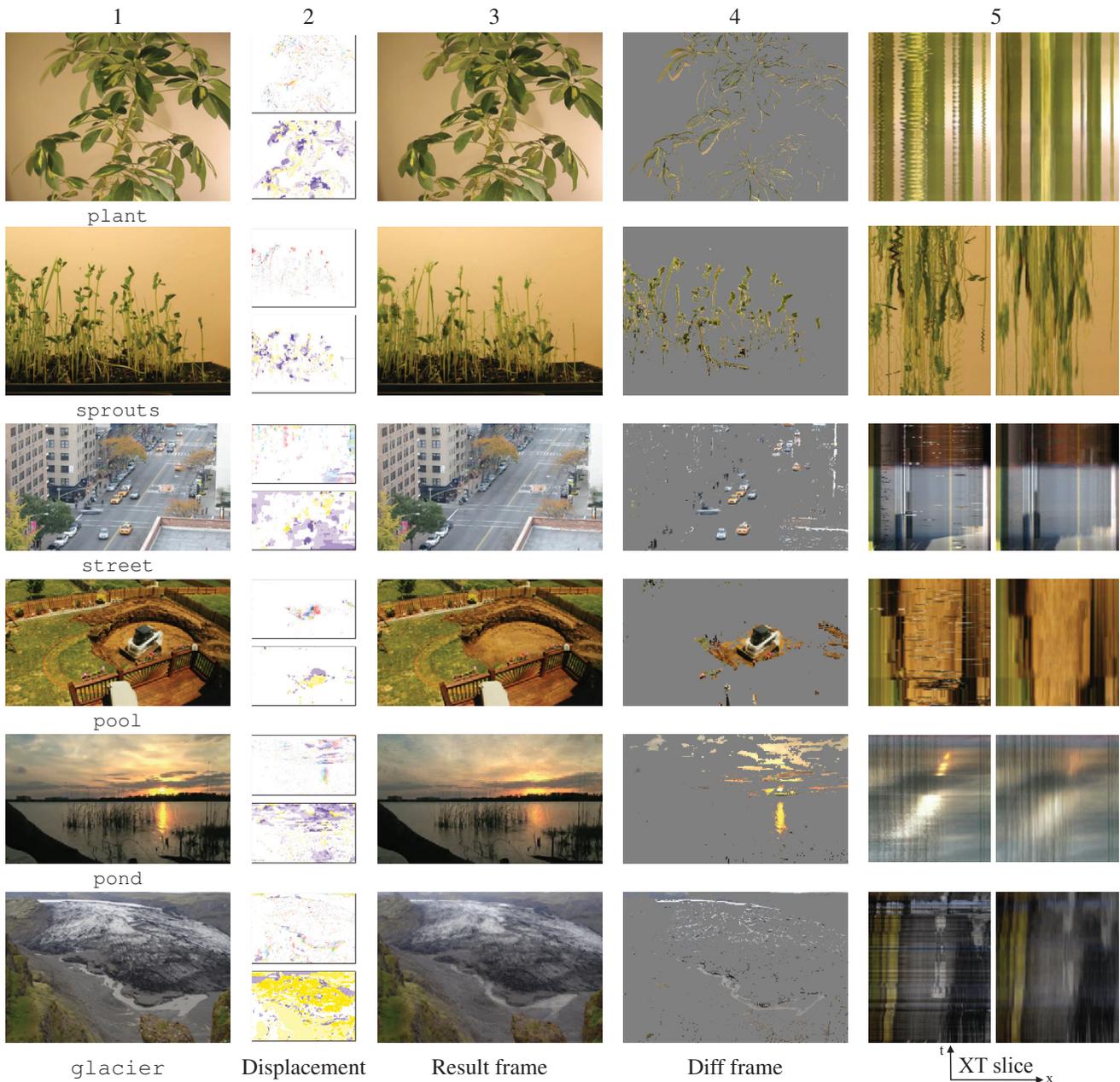


Figure 8. Motion denoising results on time-lapse videos. For each sequence, we show a representative source frame (column 1), result frame (column 3), and the computed displacement field (column 2; spatial displacement on top, temporal at the bottom). Column 4 illustrates the delta between the source and result by thresholding the color difference between the frames and copying pixels from the source. Column 5 shows a part of an XT slice of the video volumes for the input (left) and motion-denoised sequence (right).

- [19] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *TPAMI*, pages 1068–1080, 2007. 3, 4
- [20] M. Tappen and W. Freeman. Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters. In *ICCV*, pages 900–907, 2003. 4
- [21] Y. Ukrainitz and M. Irani. Aligning sequences and actions by maximizing space-time correlations. In *ECCV*, pages 538–550, 2006. 3
- [22] Y. Weiss. Deriving intrinsic images from image sequences. *ICCV*, 2:68–75, 2001. 3
- [23] J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, 8:236–239, 2003. 4