Motion Magnification

Ce Liu Antonio Torralba William T. Freeman Frédo Durand Edward H. Adelson

Computer Science and Artificial Intelligence Lab (CSAIL)
Massachusetts Institute of Technology*

(a) Input sequence (b) Motion magnified sequence

Figure 1: Frames from input and motion magnified output sequence. The algorithm groups the input (a) into motion layers and amplifies the motions of a layer selected by the user. Deformations of the swing support elements are revealed in the motion magnified output sequence (b), magnifying the original motions by a factor of 40.

Abstract

We present motion magnification, a technique that acts like a microscope for visual motion. It can amplify subtle motions in a video sequence, allowing for visualization of deformations that would otherwise be invisible. To achieve motion magnification, we need to accurately measure visual motions, and group the pixels to be modified. After an initial image registration step, we measure motion by a robust analysis of feature point trajectories, and segment pixels based on similarity of position, color, and motion. A novel measure of motion similarity groups even very small motions according to correlation over time, which often relates to physical cause. An outlier mask marks observations not explained by our layered motion model, and those pixels are simply reproduced on the output from the original registered observations.

The motion of any selected layer may be magnified by a user-specified amount; texture synthesis fills-in unseen "holes" revealed by the amplified motions. The resulting motion-magnified images can reveal or emphasize small motions in the original sequence, as we demonstrate with deformations in load-bearing structures, subtle motions or balancing corrections of people, and "rigid" structures bending under hand pressure.

Keywords: video-based rendering, computer vision, video processing, motion processing

1 Introduction

Visual motion can occur at different amplitudes, and over different temporal and spatial frequency scales. Small motions are difficult to observe, yet may reveal important information about the world: small deformations of structures, minute adjustments in an equilibrium process, or the small movements of a system in response to some forcing function. We want a machine which will reveal and clarify those motions, much as a microscope can reveal small and invisible structures.

We have developed a technique, called Motion Magnification, which acts like a microscope for motion in video sequences. The algorithm analyzes the motions of an input video sequence, allowing a user to specify a cluster of pixels to be affected, and how much their motions are to be magnified. Typically, small motions are amplified and large motions are left unchanged. The final stage of motion magnification is to render the sequence with the desired motions magnified by the specified amount.

While motion magnification is conceptually simple, performing it without objectionable artifacts requires considerable attention to detail. We introduce techniques to analyze motion robustly, verifying estimated motions as well as their regions of support. The selection of motions to be amplified is made simple and intuitive by an automatic grouping process, where a prespecified number of pixel clusters are found, based on their similarity in position, intensity, and motion characteristics. For this, we introduce a measure of affinity that groups points based on their trajectories over time, not just the similarities of their instantaneous velocities. The user specifies which cluster's motions should be amplified and by how much. Holes revealed by amplified motions are filled using texture synthesis methods.

We demonstrate motion magnification with several proof-of-concept examples, magnifying small-amplitude motions in videos of structures or people. We envision potential applications ranging from engineering diagnosis or instruction to comedic amplification of ordinary expressions.

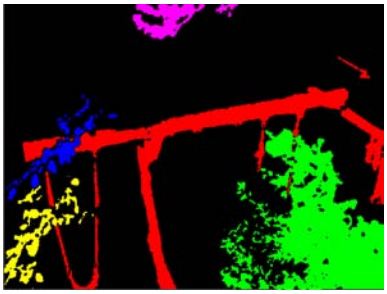
2 Related Work

Motion magnification analyzes and redisplays a motion signal, and thus relates to research on manipulating and redisplaying motion capture data, such as modifications to create new actions from others [Arikan and Forsyth 2002; Lee et al. 2002; Pullen and Bregler 2002; Li et al. 2002], and methods to alter style [Brand and Hertzmann 2000; Gleicher 1998; Unuma et al. 1995]. Of course, our problem is in a different domain; we manipulate video data, not marker positions, and thus have a more difficult analysis task,

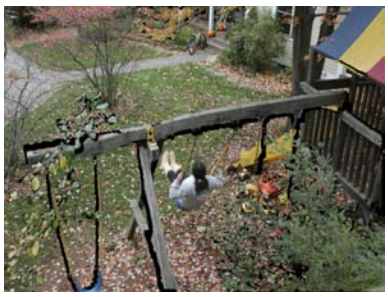
*Email: {celiu, torralba, billf, fredo, adelson}@csail.mit.edu

(a) Registered input frame

(b) Clustered trajectories of tracked features

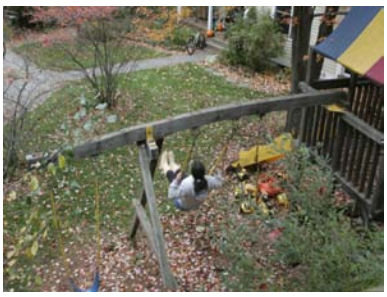


(c) Layers of related motion and appearance



(d) Motion magnified, showing holes

(e) After texture in-painting to fill holes



(f) After user's modifications to segmentation map in (c)

Figure 2: Summary of motion magnification processing steps; see overview in Section 3.

but also have the richer synthesis possibilities of video.

Several other projects have used video motion analysis in order to synthesize an output. Brostow and Essa [Brostow and Essa 2001] tracked frame-to-frame motion of objects, then integrated the scene's appearance as it changed over a synthetic shutter time to simulate motion blur. Video textures [Schodl et al. 2000] analyzes the overall similarity of all frames to identify candidate temporal jumps in a modified playback of the video. Several researchers have used a layered motion analysis to synthesize a modified video [Wang and Adelson 1994; Jojic and Frey 2001], where the modification typically involves removing individual layers. However, many of the techniques used to group pixels of similar motions into layers would not work for our difficult case of analyzing very small motions. While the above projects relate at the general level of motion analysis followed by synthesis, we are not aware of any previous work addressing motion magnification.

3 Overview

We want to find small motions in a video and magnify them. We model the appearance of the input video as trajectories (translations) of the pixel intensities observed in a reference frame. Naïvely, this sounds like one simply needs to (a) compute the translation from one pixel to the next in each frame, and (b) re-render the video with small motions amplified. Unfortunately, such a naïve approach would lead to artifactual transitions between amplified and unamplified pixels within a single structure. Most of the steps of motion magnification relate to reliably estimating motions, and to clustering pixels whose motions should be magnified as a group. While we ultimately estimate a motion at every pixel, we begin by analyzing and grouping the motions of feature points, local intensity configurations that are promising candidates for finding reliable motion trajectories. Below we motivate and summarize each step of the motion magnification processing. The processing steps are illustrated with the swing set images in Fig. 2.

3.1 Register input images

When we magnify small motions, it is essential to begin by registering the frames, to prevent amplifying the inevitable small motions due to camera shake. For this step, we assume that the input image sequence depicts a predominantly static scene. We perform an initial tracking of detected feature points and find the affine warp which best removes the motions of the set of tracked feature points, ignoring outliers. After intensity normalization for any exposure variations, the resulting registered images are ready for motion analysis.

3.2 Cluster feature point trajectories

In order that the motion magnification not break apart coherent objects, we seek to group objects that move with correlated (not necessarily identical) motions. To achieve this, we robustly track feature points throughout the sequence, then cluster their trajectories into K sets of correlated motions. One special cluster of feature points with no translation over frames is the background cluster. An important contribution of this work is the computation of trajectory correlation in a manner invariant to the overall scale of the motions, thereby allowing very small motions to be grouped with larger motions to which they are correlated. For example, the left extremity of the beam in Fig. 2 has larger motion magnitude than the points attached to the vertical beam, and some points along the beam move in opposite phase, yet, they should all be assigned to the same motion layer because they belong to a "common cause". The motions are all specified as translations from the feature point position in a reference frame.

3.3 Segmentation: layer assignment

From the clustered feature point trajectories, we want to derive motion trajectories for each pixel of the reference frame. We interpolate a dense motion field for each motion cluster, giving us K possible motion vectors at each pixel. We need to assign each pixel of every frame to one of the clusters or motion layers.

It is possible, in principle, to perform segmentation using motion alone [Wang and Adelson 1994; Jojic and Frey 2001], but reliable segmentation requires the use of additional features. We use pixel color, position, as well as motion to estimate the cluster assignment for each pixel, defining a Markov random field which we solve using graph cuts [Boykov et al. 2001]. To impose

temporal consistency, we then assign each pixel trajectory to its most commonly assigned cluster over all time frames.

This gives us a layered motion representation such as that proposed by Wang and Adelson [1994], but generalizing layer membership to include correlated motions, and not just similar ones. Our model of the video is a set of temporally constant pixel intensities, clustered into layers, which translate over the video sequence according to interpolated trajectories that are unique to each pixel. The layer ordering can be specified by the user, or computed using the methods of Brostow and Essa [1999]. In practice, it is sufficient to randomly assign the ordering of non-background layers if the magnified layer has minimal occlusions with other layers, as is often the case. At each stage, pixels which do not fit the model are relegated to a special "outlier layer". The other layer that is treated specially is the background layer. Regions of the background layer which were never seen in the original video sequence may be made visible by amplified motions of motion layers above the background. We thus fill-in all holes in the background layer by the texture synthesis method of Efros and Leung [1999].

3.4 Magnify motions of selected cluster

After the layers are determined, the user specifies a layer for motion magnification. Presently, the magnification consists of amplifying all translations from the reference position by a constant factor, typically between 4 and 40, but more general motion modification functions are possible.

3.5 Render video

Following motion magnification, we render the modified video sequence. The background layer is constant for all frames and we render its pixels first. Then the pixels assigned to the outlier layer are copied as they appeared in the registered input frames. Finally, the pixels of the remaining layers are written into the output sequence. The intensities are those of the reference frame; the displacements are those of the measured or magnified motions, as appropriate to the layer. In the following sections, we describe each processing step of motion magnification in detail.

4 Robust Video Registration

Since we magnify small motions, we need to be very careful that stationary pixels are not classified as moving. Because of inevitable small camera motions, even with a tripod, almost all pixels are moving in the input sequences. To address this problem, we devised a fully automatic system to align the input images.

The main idea comes from recent work [Sand and Teller 2004]. Instead of registering images frame to frame, our algorithm finds a reliable set of feature points that are classified as "still". Then an affine motion is estimated from the matched feature points. All frames are registered to a reference frame, typically the first frame in our system.

We detect corners at different scales in the reference frame using a hierarchical version of Harris corner detector [Harris and Stephens 1988], with a modification from page 45 of Nobel's thesis [Nobel 1989]. Then we compute a flow vector for each feature point from the reference frame to each of the other frames based on the minimum sum of squared differences (SSD) over a small patch. The precision of the flow vector is further refined to sub-pixel level based on a local Lucas-Kanade algorithm [Lucas and Kanade 1981; Shi and Tomasi 1994].

Before describing the affine motion estimation, we introduce some notation conventions of the paper. We measure N feature points over K frames. The n^{th} $(n = 1 \cdots N)$ feature point in frame k $(k = 1 \cdots K)$ is denoted as (n,k). The coordinate of feature point (n,k) is (x_{nk}, y_{nk}). Likewise, the flow vector from the reference frame is denoted as (v^x_{nk}, v^y_{nk}). For similarity measurements, we consider a window or patch B_{nk} of size $2w \times 2w$ around each feature point (n,k). $B_{nk}(p,q)$ is the pixel at relative coordinates (p,q) from the centroid of patch B_{nk}. Note that when we use sub-pixel coordinate (x_{nk}, y_{nk}), we interpolate the patch using bicubic reconstruction.

A global affine motion $A_k \in \mathbb{R}^{2\times 3}$ is estimated from the reference frame to frame k with a weight depending on the quality of the local appearance match. The probability that a feature point (n,k) participates in this affine motion is estimated as

$$\Pr_{nk} = \exp\{-\|A_k[x_{nk}\ y_{nk}\ 1]^T - [v^x_{nk}\ v^y_{nk}]^T\|^2/(2\sigma_k^2)\} \qquad (1)$$

where variance σ_k is estimated as the mean reconstruction error $\sigma_k = \frac{1}{n}\sum_n \|A_k[x_{nk}\ y_{nk}\ 1]^T - [v^x_{nk}\ v^y_{nk}]^T\|^2$. We treat the ego motion of the camera as random noise, and therefore the probability for feature point n contributing to the global affine motion over all frames is the product of the probability for each frame: $\Pr_n = \prod_k \Pr_{nk}$.

Finally, the stable feature points are selected by their probability relative to that of the most probable feature point:

$$\Pr_n > \alpha^K \cdot \max_i \Pr_i. \qquad (2)$$

We find $\alpha = 0.3$ works well for all the sequences we have tested. By this procedure, unstable feature points, such as those on occluding boundaries, in disoccluded regions and at rapidly moving objects, are discarded. Only the feature points that consistently contribute to a global affine motion across all the frames are selected for registration.

When the stable feature points are selected, we redo the SSD matching and local Lucas-Kanade refinement from the reference frame to each of the rest of the frames. The rest of the frames are all registered to the reference frame from a global affine warp $\tilde{A}_k$ estimated from the matching upon this stable feature point set. In this step we also perform histogram equalization for each frame to the reference frame to remove illumination or exposure changes.

5 Robust Computation and Clustering of Feature Point Trajectories

In the previous section, we computed feature trajectories for the static background in order to stabilize the sequence. We now turn to the computation of trajectories for feature points over the whole image and to their clustering into motions that are correlated.

5.1 Variable region feature point tracking

Once the images have been registered, we find and track feature points for a second time. The goal of this feature tracking is to find the trajectories of a reliable set of feature points to represent the motions in the video. As before, the steps consist of feature point detection, SSD matching and local Lucas-Kanade refinement. For simplicity, we use only those features detected in the first frame.

To achieve reliable feature point matching near occlusion boundaries, [Sawhney et al. 2001] displaced the rectangular region of support away from the boundary edge. Here we introduce a method to find regions of support of general shape, tailored to each feature point. We compute a weight or support map for each feature patch that characterizes how pixels should be weighted in the SSD similarity measures. For features near occlusion boundaries, this increases reliability by only comparing pixel intensities with others on the same side of the boundary. This map, shown in Fig. 3, also lets us assess the validity of each feature trajectory, useful in both the SSD matching and Lucas-Kanade refinement. We call this method *variable region feature point tracking*.

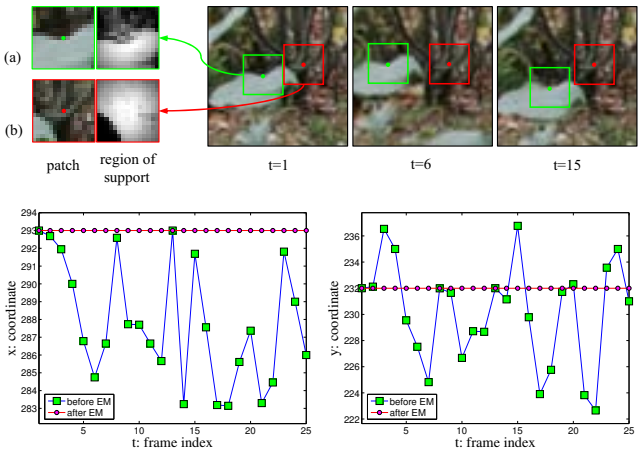


Figure 3: Learned regions of support allow features (a) and (b) to reliably track the leaf and background, respectively, despite partial occlusions. For feature (b) on the stationary background, the plots show the x (left) and y (right) coordinates of the track both with (red) and without (blue) a learned region of support for appearance comparisons. The track using a learned region of support is constant, as desired for feature point on the stationary background.

We use an Expectation Maximization (EM) algorithm [Dempster et al. 1977] to learn the weight map associated with each feature point. The EM algorithm alternates between an "E-step", when the weight map (region of support) is estimated for an assumed feature translation, and an "M-step", when the feature translation is updated, using the weight map just estimated.

E-step

We first estimate the probability that each feature point trajectory is reliable. We call reliable trajectories inliers and unreliable ones outliers. Two conditions must be met for the inliers: (1) the SSD matching error at each position of the feature point (compared with the appearance in the reference frame) must be small, and (2) there must be nearby feature point positions from other times along the trajectory. To compute the second term, we evaluate the mean distance to the N nearest feature points in the same trajectory. The tracking inlier probability of feature n at frame k ($k > 1$) is computed as

$$\Pr_{nk} = \exp\{-\frac{\mathrm{SSD}_{nk}}{2 \min_{1<i\leqslant K} \mathrm{SSD}_{ni}} - \frac{d_{nk}}{2 \min_{1\leqslant i\leqslant K} d_{ni})}\}, \qquad (3)$$

where SSD_{nk} is the SSD of feature n at frame k, and d_{nk} is the mean distance of feature n at frame k to the N-nearest feature points in the same trajectory.

The weight map is estimated from the average reconstruction error for the pixel (p,q) ($-w \leqslant p,q \leqslant w$) relative to the feature point. We use a patch size $w = 7$ for all the examples. The weight is therefore computed as

$$\Phi_n(p,q) = \exp\{-\frac{p^2+q^2}{2s^2} - \frac{\sum_{k=2}^{K} \|B_{nk}(p,q) - B_{n,1}(p,q)\|^2 \Pr_{nk}}{2\sigma_n^2 \sum_{k=2}^{K} \Pr_{nk}}\} \qquad (4)$$

where $s = w/2$ and σ_n^2 is the mean variance over frames k's and positions (p,q). Intuitively, the neighboring pixels that are close to the feature point and have less variation in matching should have high weight.

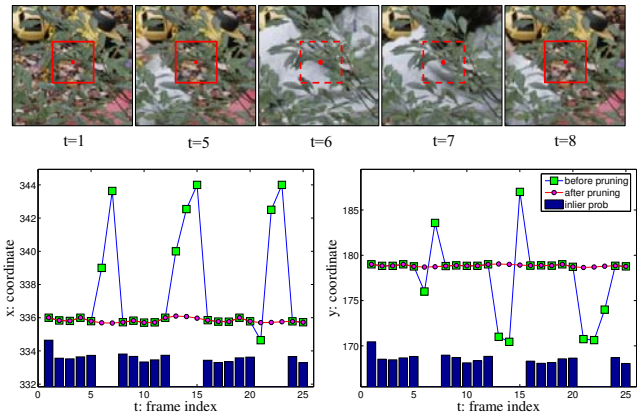


Figure 4: Top images show a feature point on the stationary background layer becoming occluded during frames 6 and 7. Below are the x- and y- coordinates of the tracked feature, showing outlier positions. These can be identified from the inlier probabilities shown as a bar plot (repeated for comparison with each plot) and replaced with smoothed values.

M-step

The M-step is the same as the previous SSD matching and local Lucas-Kanade refinement except that the weight map Φ_n is used. The use of this map, indicating the region of support for each feature, results in more reliable feature point tracking, illustrated in Fig. 3.

Experimentally, we found that after 10 iterations of EM most of the feature points converge to a reliable estimate of the weight map as well as the local flow vector. However, some feature points do not yield valid trajectories and must be pruned. We use a number of criteria to prune feature points:

- **Minimum matching error** Some feature points may appear in the reference frame but not in others. For such cases, the minimum matching error remains high. We remove these spurious trajectories by setting an upper bound on the minimum matching error; this threshold is set so that feature points which appear only in the first frame are removed.
- **Inlier probability** Some feature points reappear, but seldom, indicating an unreliable trajectory. We use the average inlier probability $\frac{1}{K}\sum_k \Pr_{nk}$ as a metric for trajectory n. A trajectory is removed if the average inlier probability is below a threshold, set to be 30% of the mean inlier probability for the trajectory.
- **Matching outlier detection, removal and filling** Finally, we must smooth some of the remaining inlier feature point trajectories because of occlusions at some frames, which generates impulsive noise in the trajectory and a corresponding increase in the matching error, as shown in Fig. 4. These events are detected from the inlier probability at each time frame and removed. We fill in the missing feature point positions by minimizing the second derivative energy of the trajectory over time, by summing the squared responses from filtering with $[-1\ 2\ -1]$. The resulting least squares problem is solved by a standard conjugate gradient method, [Strang 1986].

The first two criteria are used both before and after the EM algorithm, and the third criterion is applied only after EM. The output of the feature point tracking is a set of feature points, their regions of support, reliabilities, and trajectories over time. These trajectories are output to the next module for clustering.

5.2 Clustering by coherent motion

We seek to group related feature point trajectories into clusters, which will form the basis for our assignment of pixels to motion layers. Using motion, we can group regions of varying appearance or without spatial continuity (due to occlusions). Our goal in the clustering is that motions with a common cause be grouped together, even though the motions may be in different directions.

To do this, we use the entire motion trajectory of each feature point, not just instantaneous motions. Motions caused by the same physical source tend to covary and have common modes of resonance [Zelnik-Manor and Irani 2003]. We introduce the use of *normalized correlation* between the trajectories as a measure of their similarity. Composing the x and y components of the velocities into a complex motion vector, the correlation index $\rho_{n,m}$ between the trajectories of two feature points n and m is:

$$\rho_{n,m} = \left| \frac{\sum_k \left(v^x_{nk} + jv^y_{nk}\right)\left(v^x_{mk} + jv^y_{mk}\right)}{\sqrt{\left(\sum_k (v^x_{nk})^2 + (v^y_{nk})^2\right)\left(\sum_k (v^x_{mk})^2 + (v^y_{mk})^2\right)}} \right| \quad (5)$$

with $j = \sqrt{-1}$.

The normalized correlation between complex velocities is invariant to both the direction of the motion trajectories, and their magnitudes. In the swingset example, the motions of the beam and of the chain attached to the beam show a high correlation index even though they have very different magnitudes and are moving in different directions (because of the normalization and absolute value operations in Eq. (5)). The normalized correlation is close to zero for feature points that belong to objects with independent motions. For instance, the trajectories of points belonging to the swing structure and the blowing leaves have very small correlation when evaluated over 25 frames.

Using the normalized correlation, we construct a similarity matrix between all feature tracks, then use spectral clustering [Shi and Malik 1998] to group them into K clusters, a number selected by the user to give physically reasonable groupings. Before clustering, feature points having fewer than 5 neighbors with $\rho > 0.9$ are removed and considered as outliers. Fig. 2(c) shows the clustering result for the swing sequence using 6 clusters.

5.3 Dense optic flow field interpolation

From each group of feature tracks, we seek to interpolate a dense optic flow field over all pixels; we will then assign each pixel to one motion group to form a layered motion representation.

Because the tracked objects can be non-rigid, we interpolated using locally weighted linear regression to estimate an affine motion for the query pixel, following the approach of [Sand and Teller 2004]. To speed the computation, we apply this interpolation on a sparse lattice of every fourth pixel. Then a bicubic interpolation is applied to obtain the dense flow field for all pixels. This two-step approach reduced the complexity by one order of magnitude with little cost in precision. We denote $M^{(l)}_{ik}$ the dense flow field from frame i to frame k for layer l.

6 Segmentation: Assignment of Each Pixel to a Motion Cluster

We seek to assign every pixel to one of the motion clusters (layers). We do so using three cues: motion likelihood, color likelihood, and spatial connectivity. In the subsections that follow, we construct grouping probabilities for each cue to form a probability for a given layer assignment that we optimize by graph cuts. Finally, we impose a temporal coherence constraint to add poorly fitting pixels to the outlier layer.

6.1 Motion likelihood

The likelihood for pixel $I_k(x,y)$ at frame k to be generated by layer l is computed from reconstruction error

$$\Pr_M(I_k(x,y)|l) = \exp\{-\sum_{i=k-u}^{k+u} \frac{\|I_k(x,y) - I_i(M^{(l)}_{ik}(x,y))\|^2}{2\sigma_M^2}\}. \quad (6)$$

Where u is the number of neighboring frames and σ_M^2 is the variance. Often, motion segmentations are based on two sequential frames, but we find that a large u, such as 10, makes motion likelihood very reliable. We can compute this since we keep the trajectories of each feature point. We assign pixels of low likelihood to the outlier layer.

6.2 Color likelihood

Color information has been widely used in interactive image editing, such as [Rother et al. 2004; Ruzon and Tomasi 2000]. We also use color to help propagate the motion cue to ambiguous (flat) regions. Similar to [Rother et al. 2004], we use a Gaussian mixture model to compute the likelihood for each pixel generated by layer l

$$\Pr_C(I_k(x,y)|l) = \sum_{i=1}^{N_C} \alpha_i^{(l)} G(I_k(x,y); \mu_i^{(l)}, \sigma_i^{(l)}) \quad (7)$$

where $\{\alpha_i^{(l)}, \mu_i^{(l)}, \sigma_i^{(l)}\}$ are estimated from a previous layer assignment, and N_C is the number of mixtures (this term is only used after a first iteration of segmentation).

6.3 Spatial connectivity

We use a compatibility function to encourage layer assignment changes at spatial discontinuities of pixel intensity. We choose the following compatibility function, which is widely used [Boykov et al. 2001; Rother et al. 2004; Wills et al. 2003]

$$V(I_k(x,y), I_k(x+p,y+q), l_1, l_2) = (p^2+q^2)^{-\frac{1}{2}} \delta[l_1 \neq l_2] \cdot \exp\{-\beta \|I_k(x,y) - I_k(x+p,y+q)\|^2\} \quad (8)$$

where $-1 \leqslant p,q \leqslant 1$, or $I_k(x,y)$ and $I_k(x+p,y+q)$ are neighboring pixels. l_1 and l_2 are the label assignment to the two pixels, respectively. Parameter β is estimated as described in Rother et al. [2004].

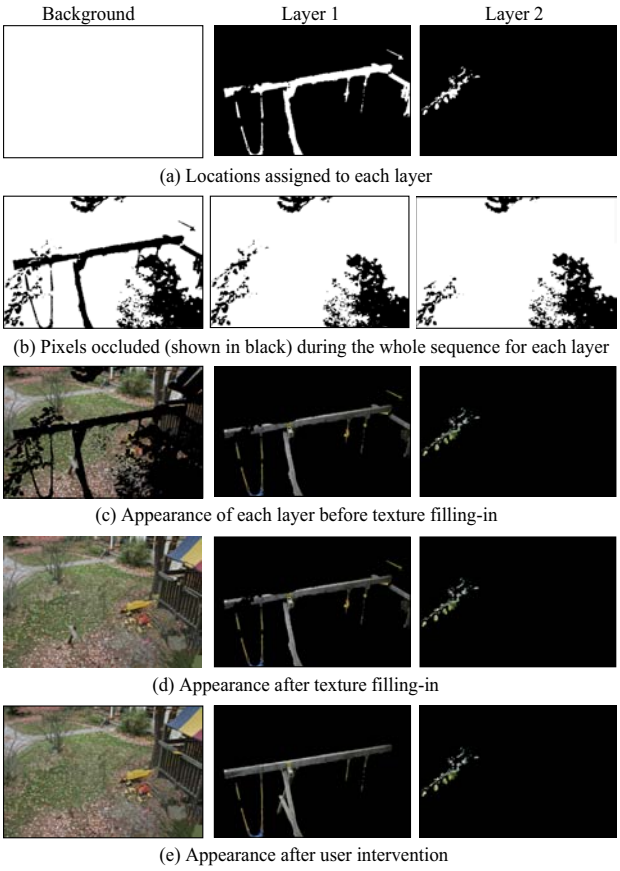
6.4 Segmentation by energy minimization

Once we have set up and learned the parameters for each of the models, we use graph cuts [Boykov et al. 2001] to minimize the total energy defined on the label assignment:

$$\begin{aligned} L^* = \arg\min_L & -\sum_{(x,y)} \log \Pr_M(I_k(x,y)|L(x,y)) \\ & -\xi \sum_{(x,y)} \log \Pr_C(I_k(x,y)|L(x,y)) \\ & +\gamma \sum_{(x,y)} \sum_{(p,q)\in N_{(x,y)}} V(I_k(x,y), I_k(x+p,y+q), L(x,y), L(x+p,y+q)) \end{aligned} \quad (9)$$

We follow Rother et al. [2004] to set $\gamma = 50$ and $\xi = 2$, which works well for our test examples.

In each iteration of the graph cut algorithm the color distribution for each layer is re-estimated. The energy function drops fastest in the first three iterations, so we applied graph cuts three times to find the layer assignment for each frame.



(a) Locations assigned to each layer

(b) Pixels occluded (shown in black) during the whole sequence for each layer

(c) Appearance of each layer before texture filling-in

(d) Appearance after texture filling-in

(e) Appearance after user intervention

Figure 5: Layered representation for the swing sequence. Only the background and two layers (out of six) are shown.

(a) (b) (c)

Figure 6: (a) Outlier locations, not well described by our model. Pixel intensities from these locations are passed through from the registered video into the rendered model (b) to yield the final composite output sequence, (c). The user specifies at which depth layer the outliers belong, in this case, above the background and below the other layers.

6.5 Final layered representation of the sequence

The energy minimization segmentation is carried out for every frame independently, which inevitably introduces changes in layer assignment from frame to frame. To build our final representation of the sequence, we project each pixel back to a reference frame using the estimated motions. Each reference frame location which projects to a motion trajectory with 80% consistent layer assignments over all frames is assigned to that layer, Fig. 5(a). (Note that reference frame pixels may be assigned to more than one motion layer). The pixel intensity for a layer at each position is set to the median of the pixel intensities assigned to that layer along the trajectory, Fig. 5(c). Since motion magnification will reveal occluded regions, we mark with occlusion masks regions where texture in-painting [Efros and Leung 1999] needs to be applied, Fig. 5(d).

Finally, we need to account for the outliers. In some sequences, outliers might correspond to important elements for which the algorithm failed to build a model. In the case of the swing sequence, the person on the swing is not tracked, due to the fast motion, and is considered as an outlier. Fig. 6(b) shows one frame of the sequence rendered without including outliers. Fig. 6(c) shows the final result when the registered input pixels from the outlier locations, Fig. 6(a), are composited into the rendered frame (above the background and below the other layers). The outlier region is the union of the outliers computed for all the frames, as described in section 6.1.

In summary, the final representation of the sequence consists in a set of N_l layers plus one outlier layer (not always required). Each layer is defined by a segmentation mask (Fig. 5(a)), and its appearance (Fig. 5(d)).

6.6 User interaction

While the automatic segmentation results are very good, the bottom-up analysis inevitably makes small errors that can lead to artifacts in the synthesized video. To address this, we allow the user to modify the layer segmentation on the reference frame, as shown in Fig. 5(d). In this example, user modifications to the segmentation and appearance maps for layer 1 removed some pixels attached to that layer in the canvas awning, completed holes in the beam, and marked the swing back leg, Fig. 5(e). Only maps in the reference frame need to be edited.

7 Magnification and Rendering

The user selects the motion layer for which the motion is to be magnified, and the displacements of each pixel in the cluster are multiplied by the selected factor. In our experiments the magnification factor was in the range between 4 and 40.

The depth ordering for each layer and the outliers can be assigned manually, or, for the non-outliers, computed through occlusion reasoning [Brostow and Essa 1999]. We render the pixels of each motion layer from back to front.

8 Experimental Results

The accompanying video shows motion magnification results for three video sequences, *handstand*, *bookshelf*, and *swingset*. The first two sequences were taken with a JVC digital HD video camera JY-HD10, which can capture images at 1086×720 resolution, 30 Hz, progressive scan. The last sequence was acquired at 8 frames per second with a Canon EOS 1D Mark II, which records images at 3500×2200 resolution. We downsample the input images for processing to 866×574 for the JVC and 1750×1100 for Canon. The precise motion computations and grouping operations of the motion magnification algorithm take 10 hours, end-to-end processing, for the swingset sequence, in a combination of C++ and Matlab code.

Handstand shows a single figure with visually perceptible balancing motions to maintain vertical posture. The magnified sequence amplifies the left-to-right corrective motion of the torso. In order to reveal small body motions without too much distortion of the human figure, we applied a saturating non-linear amplification to the motions of the magnified layer . The magnification was close to linear for amplified displacements below 40 pixels, with a compressive saturation for amplified displacements above that. Fig. 7 shows a frame from the original and motion magnified sequences. In this sequence we used a model with two layers and no outlier layer, and made no manual edits.

Bookshelf magnifies the response of a thick plywood bookshelf on aluminium supports to pressure from a hand. The original

Figure 7: The magnification result (right) for a handstand (left). The motion magnified sequence exaggerates the small postural corrections needed to maintain balance.

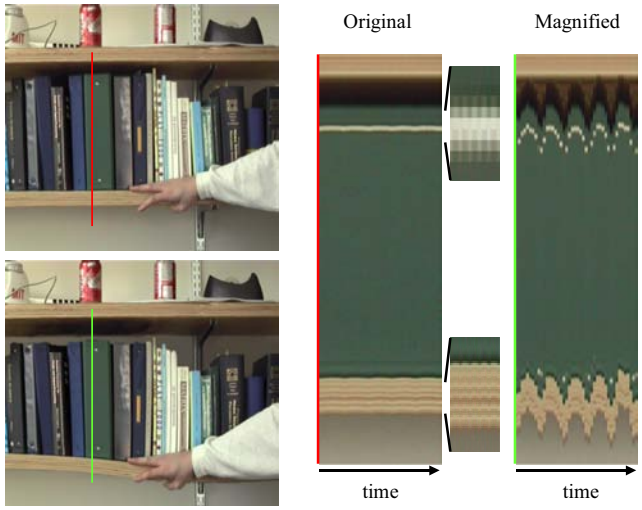


Figure 8: Under hand pressure, the bookshelf (on aluminium supports) undergoes motions that are made visible when magnified by a factor of 40 using motion magnification. User editing of reference frame masks refined the upper boundary between the books and the shelf. Also, the background layer required manual texture completion as little background is visible during the sequence.

response is barely perceptible in the original sequence, and was motion-amplified 40 times to be clearly visible in the motion magnified sequence. Fig. 8 show frames from the original and motion magnified sequences. Notice the droop of the bookshelf close to the point at which force is applied. In this sequence we used a model with two layers and no outlier layer, and made user edits as described in the figure caption.

Swingset is the most challenging sequence of the three. In this sequence we used a model with six layers and one outlier layer. The sequence has periodic motion (the swingset structure), very fast motion (the person), random motion (the leaves) all within a complicated, textured scene. The motion magnified video (stills in Fig. 1) reveals the imperceptible deformations of the swingset supports in response to the person swinging. Note that the beams and swings of the swingset are grouped into a single motion layer, based on the *correlations* of their motions, not based on the uniformity of the motions. This allows pixels with a common motion cause to be motion magnified together as a group.

Our model of the translations of each pixel over time allows us to perform post-processing steps unrelated to motion magnification, as well. To compensate for the low frame rate of the digital still camera images, we used our motion layer model of the sequence to

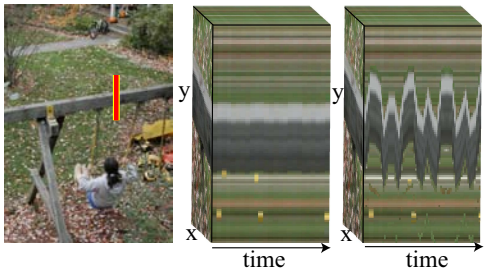


Figure 9: Section of the x,y,t volume from the original sequence (left) and the sequence after motion magnification. The detail shows a vertical section of the beam. The volume illustrates the amplification of the oscillation and also the filling of the texture behind the beam. Notice that after magnification, the motion is almost periodic despite the noise. So, the real motion of the beam is not just one single harmonic but a mixture. The original motion is amplified by a factor of 40.

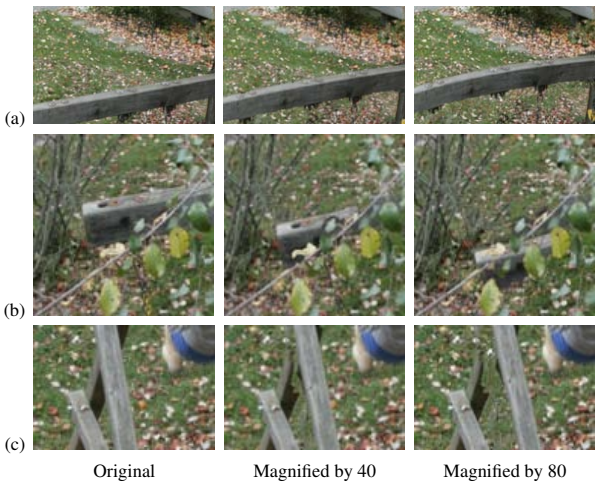


Figure 10: Details from frames of original and motion magnified swingset sequence, showing (a) beam curvature, (b) proper handling of occlusions, and (c) an artifact from imperfect automatic segmentation (before correction by the user).

interpolate missing frames, synthetically achieving a higher frame rate. For pixels of the outlier layer, we have no motion model, so we sample-and-hold replicated those pixels within the interpolated frames. This can be seen from single-stepping through the output motion magnified video, which also shows which pixels were assigned to the outlier layer.

Fig. 9 shows an x-y-t slice through part of the video volume of the swingset example, before and after motion magnification. The amplified beam motions are visible, as well as the textural filling-in of the background holes behind the displaced beam.

Fig. 10 shows details of the original and output swingset sequence, without user intervention, for motion magnifications of 40 and 80 times. Row (a) shows the bending of the support beam revealed by the magnified motion. Row (b) shows the leaf occlusions handled correctly in the synthesized output sequence. Row (c) shows a break artifact that occurred (before user editing) because the dark, occluded far leg of the swingset was not put in the same motion layer as the rest of the swingset.

9 Conclusion

We have presented a new technique, motion magnification, that reveals motions that would otherwise be invisible or very difficult to

see. The input is a sequence of images from a stationary camera. The system automatically segments a reference frame into regions of "common fate", grouped by proximity, similar color, and correlated motions. Analogous to focussing a microscope, the user identifies the segment to modify, and specifies the motion magnification factor. The video sequence is then re-rendered with the motions of the selected layer magnified as desired. The output sequence allows the user to see the form and characteristics of the magnified motions in an intuitive display, as if the physical movements themselves had been magnified, then recorded.

Acknowledgements. We acknowledge support from the National Geospatial-Intelligence Agency under contract BAA NEGI 1582-04-0004 and support from Shell Research.

References

ARIKAN, O., AND FORSYTH, D. A. 2002. Synthesizing constrained motions from examples. *ACM Transactions on Graphics 21*, 3 (July), 483–490.

BOYKOV, Y., VEKSLER, O., AND ZABIH, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Pat. Anal. Mach. Intell. 23*, 11, 1222–1239.

BRAND, M., AND HERTZMANN, A. 2000. Style machines. In *Proceedings of ACM SIGGRAPH 2000*, 183–192.

BROSTOW, G., AND ESSA, I. 1999. Motion-based video decompositing. In *IEEE International Conference on Computer Vision (ICCV '99)*, 8–13.

BROSTOW, G., AND ESSA, I. 2001. Image-based motion blur for stop motion animation. In *Proceedings of ACM SIGGRAPH 2001*, 561–566.

DEMPSTER, A. P., LAIRD, N. M., AND RUBIN, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Statist. Soc. B 39*, 1–38.

EFROS, A. A., AND LEUNG, T. K. 1999. Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision*, 1033–1038.

FISCHLER, M., AND BOLLES, R. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6, 381–395.

GLEICHER, M. 1998. Retargetting motion to new characters. In *Proceedings of ACM SIGGRAPH 98*, 33–42.

HARRIS, C., AND STEPHENS, M. 1988. A combined corner and edge detector. In *Proceedings of 4th Alvey Vision Conference*, 147–151.

JOJIC, N., AND FREY, B. 2001. Learning flexible sprites in video layers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'01)*, 199–206.

LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. 2002. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics 21*, 3 (July), 491–500.

LI, Y., WANG, T., AND SHUM, H.-Y. 2002. Motion texture: A two-level statistical model for character motion synthesis. *ACM Transactions on Graphics 21*, 3 (July), 465–472.

LUCAS, B., AND KANADE, T. 1981. An iterative image registration technique with an application to stereo vision. In *Image Understanding Workshop*, 121–130.

NOBEL, A. 1989. *Descriptions of Image Surfaces*. PhD thesis, Oxford University, Oxford, UK.

PULLEN, K., AND BREGLER, C. 2002. Motion capture assisted animation: Texture and synthesis. *ACM Transactions on Graphics 21* (July), 501–508.

ROTHER, C., KOLMOGOROV, V., AND BLAKE, A. 2004. Interactive foreground extraction using iterated graph cuts. In *Proceedings of ACM SIGGRAPH 2004*, 309–314.

RUZON, M., AND TOMASI, C. 2000. Alpha estimation in natural images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00)*, 24–31.

SAND, P., AND TELLER, S. 2004. Video matching. In *Proceedings of ACM SIGGRAPH 2004*, 592–599.

SAWHNEY, H., GUO, Y., HANNA, K., KUMAR, R., ADKINS, S., AND ZHOU, S. 2001. Hybrid stereo camera: An ibr approach for synthesis of very high resolution stereoscopic image sequences. In *Proceedings of ACM SIGGRAPH 2001*, 451–460.

SCHODL, A., SZELISKI, R., SALESIN, D., AND ESSA, I. 2000. Video textures. In *Proceedings of ACM SIGGRAPH 2000*, 489–498.

SHI, J., AND MALIK, J. 1998. Motion segmentation and tracking using normalized cuts. In *Proceedings of International Conference on Computer Vision*, 1154–1160.

SHI, J., AND TOMASI, C. 1994. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, 593–600.

STRANG, G. 1986. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press.

TORR, P. 1998. *Philosophical Transactions of the Royal Society*. Roy Soc, ch. Geometric Motion Segmentation and Model Selection, 1321–1340.

UNUMA, M., ANJYO, K., AND TAKEUCHI, R. 1995. Fourier principles for emotion-based human figure animation. In *Proceedings of ACM SIGGRAPH 95*, 91–96.

VERMA, D., AND MEILA, M. 2003. A comparison of spectral methods. Tech. Rep. UW-CSE-03-05-01, ept. of Computer Science and Engineering, University of Washington.

WANG, J., AND ADELSON, E. 1994. Representing moving images with layers. *IEEE Trans. Image Processing 3*, 5, 625–638.

WEISS, Y., AND ADELSON, E. 1994. Perceptual organized EM: A framework for motion segmentation that combines information about form and motion. Tech. rep., MIT Media Laboratory Perceptual Computing Section Technical Report No. 315.

WEISS, Y. 1997. Smoothness in Layers: Motion segmentation using nonparametric mixture estimation. In *Proceedings of Computer Vision and Pattern Recognition*, 520–527.

WILLS, J., AGARWAL, S., AND BELONGIE, S. 2003. What went where. In *Proceedings of Computer Vision and Pattern Recognition*, 37–44.

ZELNIK-MANOR, L., AND IRANI, M. 2003. Degeneracies, dependencies and their implications in multi-body and multi-sequence factorizations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, 287–293.