Video Metamorphosis Using Dense Flow Fields

Yizhou Yu Qing Wu
Department of Computer Science
University of Illinois at Urbana-Champaign
{yyz,qingwu1}@uiuc.edu

Abstract—- Metamorphosis is generally known as the continuous evolution of a source into a target. There is frame-to-frame camera or object motion most of the time when either the source or the target is an image sequence. It would be desirable to make smooth morphing transitions simultaneously along with the original motion. In this paper, we describe a novel semi-automatic morphing technique for image sequences. The proposed technique effectively exploits temporal coherence and automatic image matching. One-to-one dense mappings between pairs of corresponding frames are obtained by applying a compositing procedure and a hierarchical image matching technique based on dynamic programming. These dense mappings can be initialized by sparse frame-to-frame feature correspondences obtained semi-automatically by integrating a friendly user interface with a robust feature tracking algorithm. Experimental results show that our approach to video metamorphosis can produce superior results.

1 Introduction

Metamorphosis, or morphing is generally known as the continuous evolution of a source into a target. In many situations, there is camera or object motion, and it would be more natural to make transitions simultaneously along with the original motion than to stop the original motion while making the transitions. In this paper, we introduce a morphing technique for image sequences to achieve this effect. Typically, we assume the two input videos have the same number of frames and the natural one-to-one frame correspondence across the two sequences is actually the one we use. In addition, we also need a mapping between each pair of corresponding frames. The mappings are also referred to as *warp functions*. The animation generated from such a technique should start from the first frame of the first image sequence and end at the last frame of the second sequence. Suppose we have $N + 1$ frames in each input sequence. The frames from the first sequence are denoted as $\{F_i\}$, the frames from the second one $\{F_i{}'\}$, where $0 \leq i \leq N$. The warp function between frames F_i and F_i' is denoted as $\mathbf{W_i}$. Every frame f_i in the resulting animation should be an intermediate morph generated by $\mathbf{W_i}$.

Morphing is an artistic endeavor, the amount of required user interaction may vary with respect to the characteristics of the input videos. Semi-automatic mechanisms that allow variable user input are desirable. Such methods would also be preferred if we would like to consider video morphing as an operation in a video authoring tool that everyone can use. The temporal coherence available in image sequences can be exploited for such mechanisms. The positions of the same feature pattern in consecutive frames are likely to be close to each other. Automatic feature tracking techniques can be adapted to figure out the correspondences in this situation. Moreover, the image plane can be considered as an elastically deformable membrane. Since the amount of difference between two consecutive frames is likely to be small, a mapping between them should minimally deform the image plane so that corresponding regions from the two frames match.

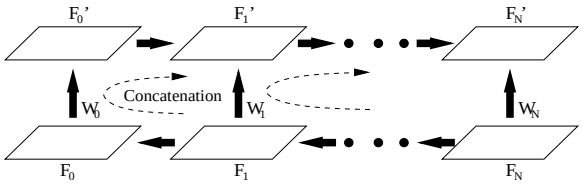


Figure 1: Dense mappings across the two input image sequences are obtained from a compositing process. This process relies on the dense mappings calculated between consecutive frames in the same sequence.

In this paper, we develop a semi-automatic framework that effectively exploits temporal coherence in the same video sequence to help establish a wide range of video morphing. A key component of this framework is to generate a complete one-to-one dense mapping between every pair of corresponding frames. To obtain the dense mapping for a single pair of frames, we develop an automatic image matching technique that incorporates orthogonal dynamic programming with image pyramids. This algorithm is first applied to obtain mappings between consecutive frames from the same sequence. Then the warp functions across two image sequences are obtained from a novel compositing process. As shown in Fig. 1, if we already know the mapping $\mathbf{W_0}$ between F_0 and F_0', the mapping between F_1 and F_0, as well as the mapping between F_0' and F_1', then $\mathbf{W_1}$ can be obtained by concatenating these three. This dense mapping algorithm will be introduced in Section 4.

The automatic image matching technique can be optionally enhanced with sparse feature correspondences to make it more robust. The sparse correspondences can be obtained either from feature tracking or from user interaction. These sparse correspondences can be used as "seeds" to guide the image matching process so that the dense mapping will be sufficiently constrained and the sparse correspondences will be satisfied as well. More details about obtaining sparse correspondences will be discussed in Section 3.

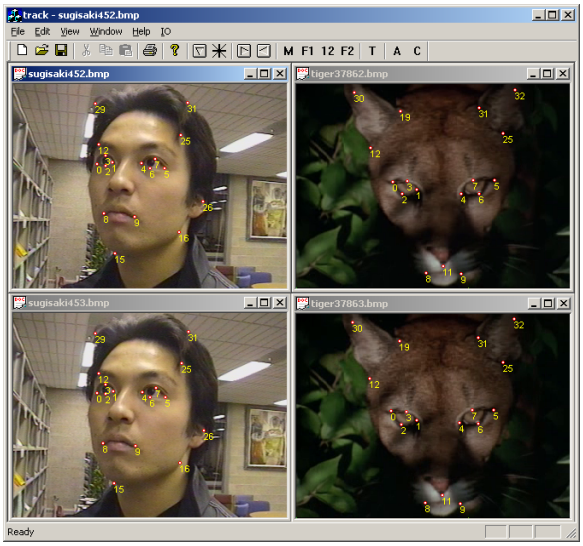
2 Related Work

Image morphing can be classified as mesh based [Wolberg 1990], or field-based [Beier and Neely 1992]. Warp function generation can be formulated as scattered data interpolation [Wolberg 1990; Ruprecht and Muller 1995] using thin-plate surface model [Litwinowicz and Williams 1994] or more general radial basis functions [Arad and Reisfeld 1995]. Enforcing a one-to-one mapping can be achieved either by expensive energy minimization [Lee et al. 1995a] or much more efficient multilevel free-form deformation across a hierarchy of control lattices [Lee et al. 1995b]. [Gao and Sederberg 1998] presents an interesting morphing algorithm for two similar images with little or no human interaction. They attempt to minimize energy terms using a fast search strategy which however, does not guarantee a good solution between two dissimilar images. Refer to the excellent surveys reported in [Wolberg 1998; Gomes

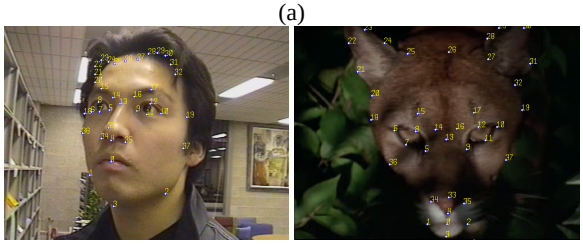
et al. 1997] for other static image morphing techniques. [Lee et al. 1998] considered the composition of mappings in warp computation.

Given that two images are sufficiently similar, image registration [Bajcsy and Kovacic 1989; Brown 1992] offers a potential solution for automatically determining the correspondence among all points across two images. Optical flow techniques [Beauchemin and Barron 1995; Quenot et al. 1998] are regularly used to track the frame-to-frame motion of video frames. Both types of techniques do not guarantee one-to-one correspondence, which is undesirable for image morphing. [Shinagawa and Kunii 1998] presents a more effective image matching algorithm using critical-point filters at multiple resolutions. Meanwhile, there are quite a few effective feature tracking algorithms [Lucas and Kanade 1981; Torresani et al. 2001; Olson 2002] for image sequences in computer vision. [Shi and Tomasi 1994] proposes criteria for choosing and updating features during tracking. [Olson 2002] describes a maximum-likelihood feature tracking strategy which is more robust than most previous methods.

Despite the large amount of work on image morphing, video morphing is much less mature although feature tracking has previously been exploited [Szewczyk et al. 1997; MorphMan 2003]. As in this paper, user assistance is necessary to correct tracking errors. In general, the tracking algorithm [Olson 2002] adopted in this paper has better performance than those in [Szewczyk et al. 1997; MorphMan 2003]. Most importantly, automatically obtaining dense mappings with temporal coherence has not previously been considered for video morphing. Previous work has not considered morphing between highly nonrigid objects such as fluids, either.



(a)



(b) (c)

Figure 3: (a) The user interface for feature tracking. Two consecutive keyframes from each sequence are displayed vertically. Four frames are shown simultaneously in the window. Each feature point is assigned an identification number which provides convenience for user verification across keyframes of the same sequence. (b)&(c) A pair of superkeyframes with their feature correspondences which include tracked features and additional user selected points.

3 Sparse Correspondences

In this section, we discuss the optional stage of obtaining sparse correspondences. The obtained sparse correspondences may be used as constraints in the following image matching to solve dense correspondences. We consider two types of sparse correspondences: the first one, called intrasequence correspondences, is between two consecutive keyframes from the same sequence; and the second one, called intersequence correspondences, is between two corresponding keyframes in two sequences. We apply an automatic and robust feature point tracking algorithm in [Olson 2002] to help with the first task. This algorithm is capable of tracking feature points with large frame-to-frame motion.

3.1 Intrasequence Feature Correspondences

The user chooses a subset of uniformly spaced frames from each input sequence as keyframes (Fig. 2). For example, there could be a keyframe every three frames. The keyframes in the two input sequences are synchronized. In practice, we track points from keyframe to keyframe assuming it is sufficiently accurate to linearly interpolate the location of a feature in an intermediate frame. In the extreme case of very large motion, every frame in the original sequence should be a keyframe and interpolation is no longer necessary. Occasionally, the user needs to adaptively adjust the locations of the keyframes for a specific feature point through user interaction.

A sparse set of significant features are automatically detected and updated for each sequence using the criteria from both [Olson 2002] and [Shi and Tomasi 1994]. The user may also interactively choose a few additional features that are crucial in delineating the shape of an object. While the features are followed automatically between keyframes, the user needs to verify the correctness of the tracking results. In the case of an error, the user can make corrections using mouse clicks. Our user study indicates that verification of the correctness of tracking can be performed four to seven times faster than clicking every feature from keyframe to keyframe. New features can be incrementally inserted conveniently when the user later finds out there is a necessity to do so.

3.2 Intersequence Feature Correspondences

Suppose there are two sets of corresponding features in the first frames of the two input sequences and each of these features can be tracked from frame to frame in its own sequence. Observing temporal coherence, the correspondences between these two sets of tracked features will be automatically maintained at every pair of frames across the two sequences. In practice, because feature tracking has accumulated errors, we need to automatically remove some of the existing features and insert new high-quality features at every keyframe [Shi and Tomasi 1994]. We also identify a sparse set of *superkeyframes* where a larger amount of user interaction can be involved besides usual feature verification (Fig. 2). The user can also interactively specify a number of additional feature correspondences to better constrain the mapping. These additional correspondences do not need to be tracked in the subsequent frames. Therefore, additional user interaction is only necessary for a very small number of superkeyframes. Nevertheless, it is more convenient for the user to edit point tracking results.

The user interface for feature tracking and a pair of su-

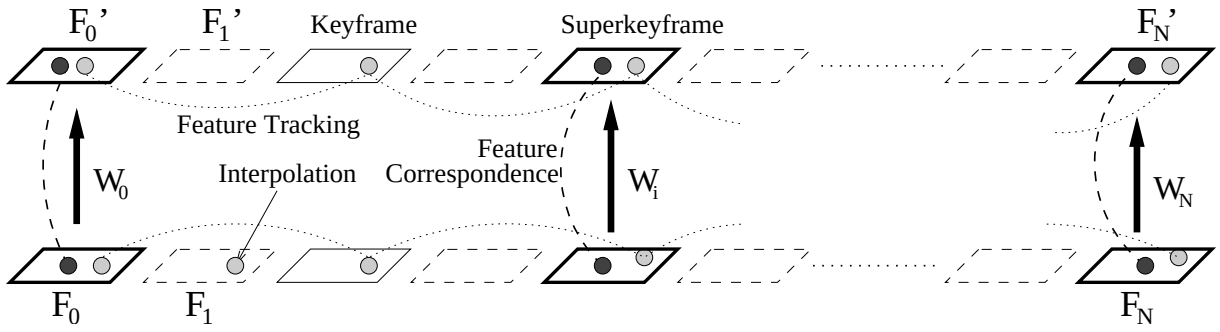


Figure 2: Sparse feature correspondences among keyframes of the same sequence are obtained using a feature tracking algorithm followed by user correction; feature locations in the intermediate frames are obtained by interpolation; feature correspondences between a pair of corresponding superkeyframes across the two sequences are obtained by exploiting temporal coherence and user interaction.

perkeyframes with marked feature correspondences are shown in Fig. 3. Fig. 3(a) shows the features we actually tracked for the puma and human face sequences. Fig. 3(b) shows the correspondences at superkeyframes of these sequences. It can be seen that Fig. 3(b) has a larger number of features than Fig. 3(a) because we sometimes need a relatively larger number of constraints for mappings across two sequences since the objects in different sequences might have very different shapes and colors.

4 Dense Mappings

A pixelwise dense mapping between two corresponding images is required for morphing. Inspired by image matching in computer vision [Bajcsy and Kovacic 1989; Shinagawa and Kunii 1998], we designed a new algorithm for generating dense mappings in a constrained optimization framework. The integration between this image matching algorithm and sparse correspondences obtained from feature tracking will be discussed in Section 4.4.

4.1 Problem Formulation for Optimal Mappings

The basic principle in automatic image matching is that we assign a score to any valid mapping between a pair of images using a cost function and an optimal mapping corresponds to a minimum score. It is desirable to minimize the amount of change in terms of shape and color in a smooth morph sequence. Therefore, our optimization formulation is based on a tradeoff between these two cues. Suppose we represent the two components of $\mathbf{W}(x,y)$ as $(U(x,y),V(x,y))$. The color difference between two points, (x,y) and $\mathbf{W}(x,y)$ is defined as the color difference between two local neighborhoods centered at the two points,

$$C_{xy}(\mathbf{W}) = \sum_{i=x-w}^{x+w} \sum_{j=y-w}^{y+w} ||\mathbf{clr}(i,j) - \mathbf{clr}(U(i,j),V(i,j))||^2 \quad (1)$$

where $\mathbf{clr}(i,j)$ is the color vector of the pixel (i,j). In practice, histogram equalization[Heeger and Bergen 1995] is applied to each color channel of the input images to align their intensity ranges.

We consider the image plane as a deformable membrane. The amount of shape change can be measured by the total amount of local deformation. The amount of local deformation at pixel (x,y) is defined as

$$\begin{aligned} D_{xy}(\mathbf{W}) &= ||\mathbf{d}_{x,y} - \mathbf{d}_{x+1,y}||^2 + ||\mathbf{d}_{x,y} - \mathbf{d}_{x-1,y}||^2 \\ &\quad + ||\mathbf{d}_{x,y} - \mathbf{d}_{x,y+1}||^2 + ||\mathbf{d}_{x,y} - \mathbf{d}_{x,y-1}||^2. \end{aligned}$$

where $\mathbf{d}_{x,y} = (U(x,y)-x, V(x,y)-y)$ is the *displacement vector* at pixel (x,y).

To evaluate the optimality of a warp function, one needs to consider the total amount of color difference and distortion over all pixels in the source image. The total cost of a warp function is defined as

$$E(\mathbf{W}) = \sum_{x,y} E_{xy}(\mathbf{W}) \quad (2)$$

where

$$E_{xy}(\mathbf{W}) = \omega_{xy}(C_{xy}(\mathbf{W}) + \mu_{xy}(\mathbf{W})) + \lambda D_{xy}(\mathbf{W}) \quad (3)$$

where ω_{xy} represents the relative importance of pixel (x,y), $\mu_{xy}(\mathbf{W})$ represents the amount of additional bias that should be imposed at a pixel for warp function $\mathbf{W}$, and λ adjusts the relative importance between color similarity and geometric distortion. We set up ω_{xy} to be proportional to the edge strength, measured by the response to an edge detector [Canny 1986], at pixel (x,y) so that matching errors at edge pixels are minimized with high priority. Typically $\mu_{xy}(\mathbf{W}) = 0$. But it is quite useful in some special situations, such as interpolating warp constraints that will be discussed in Section 4.4. λ is the only parameter that is set manually. In practice, a typical range for λ is between 0.1 and 1 if we normalize the range of each color channel. Smaller values of λ allows more geometric distortion and the image objects appear less rigid. Therefore, the lower range of λ is reserved for images with elastic or fluid-type objects while the higher range is for more rigid objects.

We seek an optimal warp function that minimizes the cost function $E(\mathbf{W})$. The warp function may be subject to two additional types of constraints. Since the above cost function alone cannot completely capture the semantics in the images, sparse warp constraints from user interaction or tracking are often necessary. Our optimization technique should be able to incorporate such constraints. The specific strategies will be discussed in Section 4.4. It is also desirable for a warp function to be a one-to-one mapping between two image planes to prevent the warped image from folding back upon itself. The second type of constraints on the warp function can guarantee such a mapping.

4.1.1 Constraints for One-to-One Mappings

Each image plane is considered as a continuous plane with subpixel accuracy. Since pixels have integer coordinates, they partition this continuous image plane into a regular grid. Suppose the warp function is $\mathbf{W}$. It can be easily shown that a one-to-one mapping exists between the rectangle defined by four neighboring pixels, (x,y), $(x+1,y)$, $(x+1,y+1)$ and $(x,y+1)$, in the source image and the quadrilateral defined by the mapped points, $\mathbf{W}(x,y)$, $\mathbf{W}(x+1,y)$,

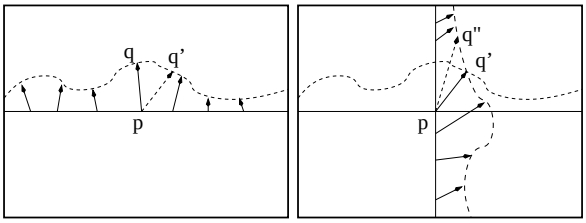


Figure 4: The horizontal and vertical passes of orthogonal dynamic programming. (a) A row of pixels(solid) in the first image corresponds to a sequence of points lying on a curve(dashed) in the second image. Suppose p originally corresponds to q. After a horizontal pass, it may correspond to a new position q' on the curve. (b) A column of pixels passing p corresponds to a second curve in the second image. After a vertical pass, p may correspond to a new position q" on the second curve.

$\mathbf{W}(x+1, y+1)$ and $\mathbf{W}(x, y+1)$, in the target image *if and only if* the target quadrilateral is not self-intersecting. To prevent the whole source image plane from being flipped when mapped to the target image plane, we also impose that every four mapped corners, $\mathbf{W}(x, y)$, $\mathbf{W}(x+1, y)$, $\mathbf{W}(x+1, y+1)$ and $\mathbf{W}(x, y+1)$ must still follow a counterclockwise order. A nonself-intersecting quadrilateral should satisfy either of the following two conditions: *i)* $\mathbf{W}(x+1, y)$ *and* $\mathbf{W}(x, y+1)$ *do not lie on the same side of the line defined by* $\mathbf{W}(x, y)$ *and* $\mathbf{W}(x+1, y+1)$*; or ii)* $\mathbf{W}(x, y)$ *and* $\mathbf{W}(x+1, y+1)$ *do not lie on the same side of the line defined by* $\mathbf{W}(x+1, y)$ *and* $\mathbf{W}(x, y+1)$. Note that a quadrilateral may become degenerate but still not self-intersecting when two adjacent corners coincide.

4.2 Iterative Orthogonal Dynamic Programming

An unknown warp function has an unknown displacement vector at every pixel. Minimizing (2) is extremely hard because of the sheer number of unknowns. We choose to optimize the warp function in an iterative framework where an updated warp function is generated every iteration based on the one obtained from the previous iteration. Thus, we generate a sequence of warp functions, $\mathbf{W_0}$, $\mathbf{W_1}$,...,$\mathbf{W_m}$, such that $E(\mathbf{W_i}) \leq E(\mathbf{W_{i-1}})$. We stop when the total cost stops decreasing or the total number of iterations has reached a predefined upper bound.

The founding component of an iteration is a one-dimensional dynamic programming scheme. For the time being, let us focus on the i_0-th row of the image. The sequence of pixels on this row are represented as $\{\mathbf{p_{i_0 j}} = (i_0, j)\}$ with fixed i_0 and varying j. Assume the current version of warp function is $\mathbf{W_c}$ which defines the sequence of mapped points, $\{\mathbf{q_{i_0 j}} = \mathbf{W_c}(\mathbf{p_{i_0 j}})\}$. The current warp function can be improved by using dynamic programming to obtain an optimal matching between these two sequences. Dynamic programming allows $\mathbf{p_{i_0 j}}$ mapped to a point other than $\mathbf{q_{i_0 j}}$ in the second sequence. The cumulative cost between a pair of points from the two sequences is defined as

$$C^{dp}(p_{i_0 j}, q_{i_0 k}) = E_{i_0 j}(\mathbf{W_c^{i_0(j \to k)}}) + \min_{l \in S_{jk}} C^{dp}(p_{i_0(j-1)}, q_{i_0 l}) \tag{4}$$

where $|j-k|$ and $|(j-1)-l|$ are bounded by the size of a search window; $\mathbf{W_c^{i_0(j \to k)}}$ is a modified warp function that maps $\mathbf{p_{i_0 j}}$ to a different point $\mathbf{q_{i_0 k}}$ in the second sequence; $E_{i_0 j}(\mathbf{W_{i_0(j \to k)}})$ follows the definition in (3); and S_{jk} contains all the admissible values of l. If the conditions for a one-to-one correspondence are not violated when $\mathbf{p_{i_0 j}}$ is mapped to $\mathbf{q_{i_0 k}}$ and $\mathbf{p_{i_0(j-1)}}$ is mapped to $\mathbf{q_{i_0 l}}$, then $l \in S_{jk}$. Once the cumulative cost for the last pixel in the row is calculated, one can trace back and find the path of the optimal matching between the two sequences.

One complete iteration in our method involves a horizontal pass and a vertical pass (Fig. 4). The horizontal pass modifies the previous warp function row by row in a scanline order using the above one-dimensional dynamic programming while the vertical pass modifies the warp function column by column in the same manner. It is not important which pass should be performed first. This process of interleaving horizontal and vertical passes are therefore referred to as *orthogonal dynamic programming*. It can move a point to anywhere in the image plane after a finite number of iterations. Since our dynamic programming scheme updates an entire row or column simultaneously, it is less likely to be stuck in local minima than locally perturbing one pixel at a time. Since our method maintains a one-to-one mapping during each iteration, it also needs such a mapping as an initialization. Fortunately, the identity warp function satisfies the conditions when there are no warp constraints.

4.3 Multiresolution

We are dealing with an extremely high dimensional optimization problem which indicates that there are a large number of local minima. The noise and high-frequency details in the original images make this situation worse. Multiresolution image pyramids have been frequently applied to image processing tasks. They have a few advantages: i) noise and high-frequency details can be removed at lower resolutions; ii) there are fewer local minima at lower resolutions; iii) it is much more computationally efficient if an approximate solution is obtained at low resolutions first.

Linear low-pass filters commonly used for image pyramids [Burt and Adelson 1983] actually do not preserve important semantic features, such as the eyes and mouth on a human face, at the same time when they remove noise. The color of a single pixel at a top level of the pyramid is the weighted average of the colors in a large region, which can potentially be larger than a semantic feature, in the original image. On the other hand, the critical point filters adopted in [Shinagawa and Kunii 1998] can effectively preserve important features such as local maxima, local minima and saddle points in lower resolution images. There are four filtered images at each level of a pyramid, one for local maxima, one for local minima and the other two for saddle points.

Inspired by the advantage offered by the critical point filters, we build a *hybrid* image pyramid for each input image. As usual, the image resolution is halved every time one proceeds to a higher level in the pyramid. The top levels are constructed using critical point filters because they can preserve important landmarks at low resolutions where low-pass filters only blur features and make different image regions look similar and ambiguous. However, when the image resolution is relatively high, the original color patterns in the image still have the most accurate information. Therefore, the lower levels are built in the same way as a Laplacian pyramid [Burt and Adelson 1983] except that the color of every pixel is represented explicitly using the three channels for the YUV color space. At top levels, the color vector has three identical channels representing the filtered luminance value obtained from the Y-channel of the lower level. In practice, the two halves of the pyramid typically interface at the middle level.

The orthogonal dynamic programming process optimizes the warp function at a single resolution. To incorporate this process with our hybrid image pyramids, there is one additional detail we still need to be careful about. Once a one-to-one mapping, $\mathbf{W^l}$, is obtained at a coarser resolution, it should be utilized to initialize such a mapping, $\mathbf{W_0^{l+1}}$, at the next higher resolution. We adopt the following scheme to achieve this goal.

- $\mathbf{W_0^{l+1}}(2x, 2y) = \mathbf{W^l}(x, y)$;
- $\mathbf{W_0^{l+1}}(2x + 1, 2y) = \frac{\mathbf{W^l}(x,y)+\mathbf{W^l}(x+1,y)}{2}$ and $\mathbf{W_0^{l+1}}(2x, 2y + 1) = \frac{\mathbf{W^l}(x,y)+\mathbf{W^l}(x,y+1)}{2}$;
- If $\mathbf{W^l}(x+1, y)$ and $\mathbf{W^l}(x, y+1)$ do not lie on the same side of the diagonal between $\mathbf{W^l}(x, y)$ and $\mathbf{W^l}(x + 1, y + 1)$, $\mathbf{W_0^{l+1}}(2x + 1, 2y + 1) = \frac{\mathbf{W^l}(x,y)+\mathbf{W^l}(x+1,y+1)}{2}$; otherwise, $\mathbf{W_0^{l+1}}(2x + 1, 2y + 1) = \frac{\mathbf{W^l}(x+1,y)+\mathbf{W^l}(x,y+1)}{2}$.

4.4 Interpolating Warp Constraints

In many situations, there exist constraints that the optimized warp function is required to satisfy. The constraints can be expressed as a sparse set of correspondences involving the unknown warp function, $\{\mathbf{W}(x_i, y_i) = (x_i', y_i')\}$. The following steps can be taken to obtain the desired warp function.

- Align these two point sets using a global affine transformation which has a closed form linear least-squares solution [Bovik 2000]. Any global translation and/or rotation between these two point sets should be compensated by this transformation. However, each pair of transformed points may still have a residual displacement between them.
- Use a scattered data interpolation scheme to interpolate the residuals from the previous step. Basically, one mapping constraint gives rise to two equations: $U(x_i, y_i) = x_i'$ and $V(x_i, y_i) = y_i'$. Two interpolating functions should be solved for these equations. The multiresolution FFD method presented in [Lee et al. 1995b] is an example suitable for this purpose since it generates a one-to-one mapping.
- Warp the target image using the interpolating mapping from the previous two steps to satisfy the sparse correspondences. Our multiresolution optimization is then performed on the original source image and the warped target image. To maintain those already aligned correspondences, one needs to assign a large negative number to $\mu_{x_i y_i}(\mathbf{W})$ in $E_{x_i y_i}(\mathbf{W})$ when $\mathbf{W}(x_i, y_i) = (x_i', y_i')$. If the constrained correspondences have small errors themselves, one would like to approximate the constraints instead of interpolation. Then, $\mu_{x_i y_i}(\mathbf{W})$ should be kept negative when $\|\mathbf{W}(x_i, y_i) - (x_i', y_i')\|$ remains close to zero. A large negative number cancels all or part of the positive cumulative cost from the rest of the pixels and keeps the final cost favorable in dynamic programming.

5 Morph Sequence Generation

5.1 Mapping Image Sequences

We need to obtain a dense mapping between every pair of corresponding frames from two image sequences. Suppose a warp function $\mathbf{W_i}$ has been obtained between frames F_i and F_i'. Applying the image matching algorithm in the previous section with or without sparse correspondences, we can obtain a warp function $\mathbf{W_i^{intra}}$ between F_i and F_{i+1}. Obtain another warp function $\mathbf{W_i^{intra'}}$ between F_i' and F_{i+1}' similarly. Then the warp $\mathbf{W_{i+1}}$ between frames F_{i+1} and F_{i+1}' can be obtained by compositing the following three warps: *the inverse of* $\mathbf{W_i^{intra}}$, $\mathbf{W_i}$, *and* $\mathbf{W_i^{intra'}}$. The inverse of $\mathbf{W_i^{intra}}$ can be easily obtained by adopting a reversed mapping direction from F_{i+1} to F_i. This relationship between $\mathbf{W_i}$ and $\mathbf{W_{i+1}}$ is shown in Fig. 1 and can be formulated as

$$\mathbf{W_{i+1}}(\mathbf{p}) = \mathbf{W_i^{intra'}}(\mathbf{W_i}(\mathbf{W_i^{intra^{-1}}}(\mathbf{p}))). \quad (5)$$

Because $\mathbf{W_i^{intra'}}$ and $\mathbf{W_i^{intra^{-1}}}$ encode the coherent frame-to-frame motion between consecutive frames, the above composite mapping can effectively preserve temporal coherence among the warp functions, $\{\mathbf{W_i}\}$. An initial warp function for the first pair of frames is necessary to start the whole process. Because of accumulated errors, we should recalibrate the process a few times at intermediate superkeyframes. The inverse mapping of $\mathbf{W_i}$ can also be obtained similarly.

Our method needs directly calculated intersequence dense mappings at a very sparse set of superkeyframes which always include the first frame. The dense mappings at intermittent superkeyframes serve as calibrating mappings since the above procedure can accumulate errors from frame to frame. These mappings can be solved using our image matching algorithm. The procedure outlined in Section 4.4 needs to be followed when there are sparse feature correspondences as warp constraints. These feature correspondences are obtained with the help of the user, as mentioned in Section 3.2. Once a superkeyframe is reached, the differences between the mapping obtained from composition and the mapping calculated directly are uniformly distributed across all the frames between the current and previous superkeyframes.

Our composite mappings can perform better than mappings obtained independently for each pair of frames because they maintain temporal coherence more effectively. They ensure that the incremental changes in the mapping are accumulated only from the small frame-to-frame motion in each individual sequence while independently obtained mappings may have relatively large discontinuities. This has been confirmed in our various experiments.

5.2 Transition Functions

Transition control tries to determine the rate of warping and color blending across a traditional morph sequence between two images [Wolberg 1998]. The transition rate can vary at different parts of the image plane. Therefore, a transition function $T(t, \mathbf{p})$ is defined over time as well as the spatial location on the image plane. The most commonly used linear transition function can be written as $T(t, \mathbf{p}) = t$. A specially designed nonuniform transition function is able to transform different regions of the source image at different rates in order to achieve certain dramatic visual effects [Lee et al. 1995a].

For video morphing, each intermediate frame of a morph sequence is generated from a different pair of input frames. However, there is only one transition function controlling the whole morph sequence. The intermediate frame f_i can be generated using $T(\frac{i}{N}, \mathbf{p})$. In a more general situation, the transition of shape and color does not have to be synchronous. We can adopt two distinct transition functions, T_g and T_c, for controlling the transition rate of shape and color separately. This strategy can create interesting color transfer effects. For example, an image object may still keep the shape of a source object while its color already becomes that of the target object. Such an effect can help visualize the underlying geometric distortion introduced by the warp function. An example of asynchronous transition of shape and color is given in Fig. 6(b).

The final images of a video morph are generated by quadrilateral rendering using OpenGL [Wolberg 1990].

6 Experimental Results

We have implemented the video morphing technique and performed extensive experiments with the system. In this section we report

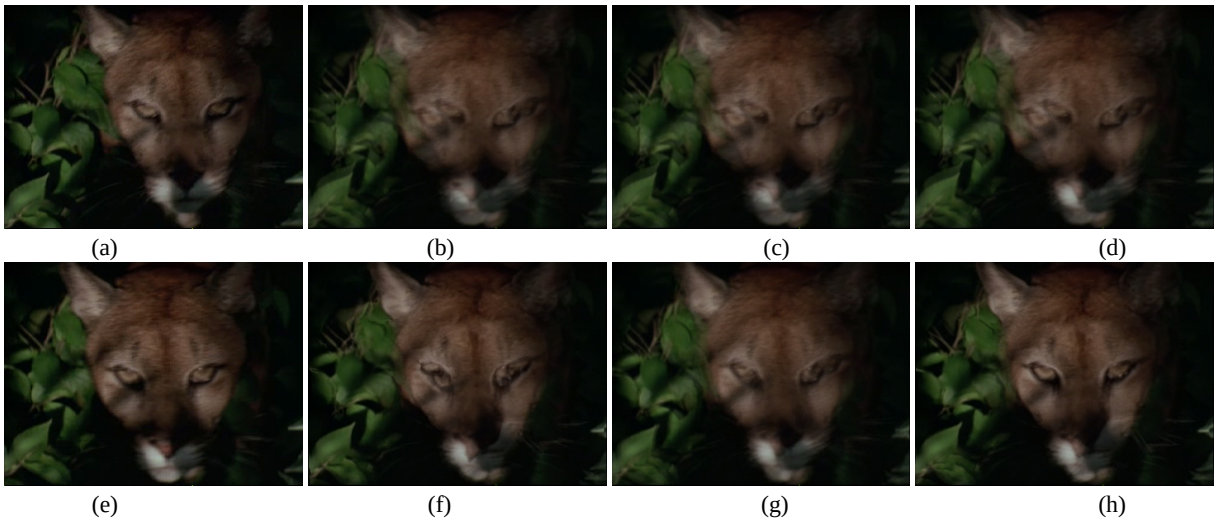


Figure 5: Comparisons among various algorithmic combinations. (a)&(e) Two frames from an image sequence. They are 10 frames apart in the original sequence. (b) A superimposed image generated from (a)&(e) to show their discrepancy. (c) The matching result obtained from only applying orthogonal dynamic programming at the highest resolution. No image pyramid is used. (d) The matching result from a Laplacian pyramid. (f) The matching result from a pyramid constructed using critial point filters. (g) The matching result from a hybrid pyramid and pixelwise 2D local search. (h) The matching result from a hybrid pyramid and dynamic programming, which produced a more accurate one-to-one mapping than the other methods. The matching between corresponding eyes is perfect. In this comparison, there are no warp constraints, and no feature tracking is performed. ©1988 NGT, Inc.

results on a variety of image objects and motion. On a pair of 60-frame image sequences at video resolution, automatic feature tracking and image matching take between 1.5 to 3 hours on a Pentium IV 2GHz PC while the total amount of time for user interaction is less than 2 hours if one chooses to have a keyframe for tracking in every two frames. After that, the user can still choose to make minor changes if the results are not completely satisfactory. The machine needs to go through image matching and morph sequence production after the user makes changes.

6.1 Comparisons

The first comparison concerns the performance of our automatic image matching algorithm. We chose two frames from an image sequence. The two frames have the same objects (an animal and leaves), but are ten frames apart in the original video. Although there are large displacements between corresponding features, it is straightforward to verify the correctness of the resulting matching. Besides linear time dynamic programming, another efficient optimization method is pixelwise local search which sequentially optimizes the correspondence at each pixel instead of updating one complete row or column of pixels simultaneously. Fig. 5 shows the results from various algorithmic combinations. Our algorithm produces a more accurate mapping than other methods. Dynamic programming can perform better than local search because it is less likely to be stuck in a local minimum. However, performing dynamic programming only at the highest resolution is ineffective because of the huge number of unknowns and the presence of noise while integrating it into each level of the pyramid significantly improves the results. Our hybrid pyramids also prove to be better than pyramids constructed from Laplacian or critical point filters only. There are two major reasons why only applying critical point filters does not generate as good results. First, color images have to be converted to grey-scale images before critical point filters can be applied since maxima and minima are not well-defined among colors but among luminance values. Second, if a noisy pixel happens to be a local maximum or minimum, it can be pushed up to the top level of the pyramid and give rise to erroneous correspondences there. Low-pass filtering the original image at lower levels of the pyramid can remove noise and prevent this possibility.

The second comparison concerns the morphing between a person and a puma both of which have their own motion. There are moving shadows cast by the leaves on the face of the puma. Morphing is focused on their faces by only tracking features in the facial regions. The backgrounds are naturally cross dissolved. In addition to feature tracking, there is a superkeyframe every twelve frames with an extra superkeyframe at the end. That results in six superkeyframes for the 60-frame sequences. A linear motion model assumes the position of a feature on an intermediate frame can be linearly interpolated from the positions of the same feature in the two closest keyframes. Assuming a linear motion model between two consecutive superkeyframes is insufficient since our superkeyframes are too sparse. On the other hand, feature tracking along with very sparse superkeyframes can produce good results. We show a comparison between the results from our approach and the results from a linear motion model in Fig 7. The linear motion model produces obvious misalignments in Fig 7(b). To produce better results with this model, the user has to mark features and correspondences in a much larger set of frames.

6.2 Morphing Examples

A few video morphing examples are given in Fig. 6, Fig. 7 and Fig. 8 along with the λ values we use for Eq. 3. The examples include rigid and nonrigid objects as well as fluids. Feature tracking was performed on most of the sequences, but not all.

One of the major advantages of our method is that it can produce very interesting results for highly nonrigid objects such as fluids. Such nonrigid objects have fast nonlinear motion and volatile feature points. Marking features and correspondences becomes extremely hard by both humans and machines. When morphing such objects, correspondences across superkeyframes and feature tracking between consecutive keyframes become optional and only a very small number of points are tracked if tracking is actually per-

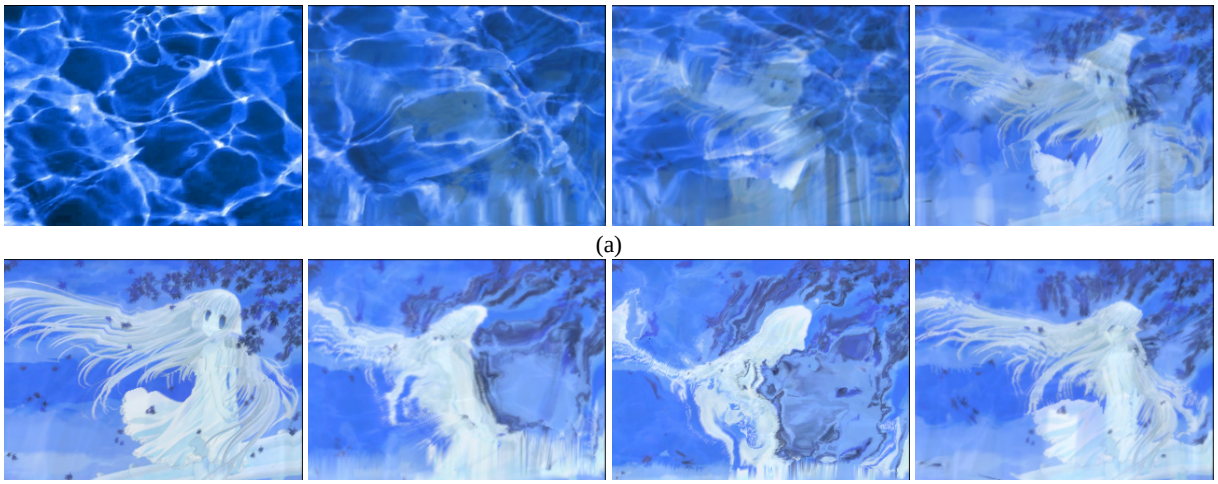


Figure 6: A video morph sequence between a pool of water and a cartoon figure. The cartoon figure deforms along with the motion of the water. (a) A morph produced using a linear transition function for both pixel location and color. (b) A morph produced using two different nonlinear transition functions for pixel location and color, respectively. The underlying warp function can be more clearly seen in this second example. $\lambda = 0.1$. ©Clamp/Kodansha

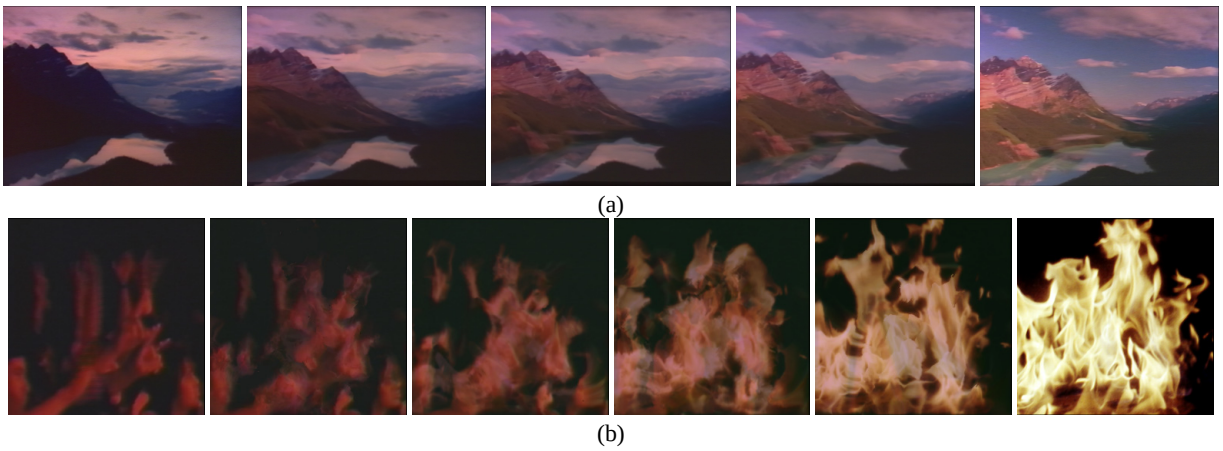


Figure 8: (b) A morph between two videos of the same landscape at different times of the day with dramatically different sky illumination. $\lambda = 0.5$. ©David Fortney (c) A video morph between an audience with waving hands and a fire sequence. $\lambda = 0.1$.

formed. However, our image matching algorithm can successfully find correspondences between images without warp constraints. For example, we found it was almost impossible to interactively mark feature correspondences between consecutive frames of the fire sequence used for Fig. 8(b), and feature tracking was not very successful, either. For the same reason, no feature correspondences across superkeyframes were generated for the examples shown in Fig. 6, Fig. 8(b) and the clouds in Fig. 8(a). However, the morph images in the middle of Fig. 8(b) show an impressive alignment between the deformed hands and fire flames. Morphing between dynamic water and a static cartoon figure shown in Fig. 6 is another difficult example. Our algorithm solved correspondences between the frames of the water and the cartoon figure automatically. We can see that the cartoon figure deforms along with the motion of the water to indicate that they are not simply cross dissolved.

Eq. (1) tries to match pixels with similar colors after histogram equalization. When the colors from two images are completely different, it still tries to match the closest ones. In Fig. 8(b), it correctly matches the hands with the flames because their colors become closest after histogram equalization. This becomes a minor issue for consecutive frames from the same sequence since the color of the same object is not expected to change much in two consecutive frames. This kind of color coherence is actually exploited in the compositing procedure for warp functions across two sequences. We only directly compute the mappings across two sequences at the superkeyframes where the user can impose a relatively larger number of warp constraints as illustrated in Fig. 3(b)-(c). These mappings are propagated to the rest of the frames using mappings between consecutive frames from the same sequence.

The video for the morphing examples in this report may be found at: `http://www-sal.cs.uiuc.edu/~yyz/movie/VideoMorph_mpeg4.avi`

7 Conclusions and Discussions

We have presented an effective morphing technique for image sequences. An important component of the approach is an automatic image matching algorithm based on hybrid pyramids and orthogo-

Figure 7: (a) a video morph using feature tracking plus superkeyframes. (b) two intermediate morph images assuming linear motion between superkeyframes. They correspond to the middle row in (a). The linear motion model produces obvious misalignment because of the sparsity of the superkeyframes. $\lambda = 1.0$.

nal dynamic programming. When automatic matching cannot capture the high-level feature correspondences in the images, this algorithm is capable of incorporating sparse warp constraints from user intervention or feature tracking. Sparse frame-to-frame feature correspondences can be obtained semi-automatically by integrating a friendly user interface with a robust feature tracking algorithm. To obtain high-quality mappings across two image sequences, a novel compositing procedure has been designed to observe temporal coherence and increase the robustness. Experimental results showed that our method performed well on a wide variety of image sequences.

A number of topics remain to be explored. Different videos may have drastically different object motion, which means that video morphing may perform better on a subset of image sequences than on the others. The video matting technique [Chuang et al. 2002] can be used for semi-automatic object and contour tracking and extraction so that one can morph a foreground object without affecting the background. We would like to extend our system so that it better supports a change of occlusion and order between multiple image objects by incorporating a multi-layer model for image and video morphing. Nevertheless, the method in this paper can still produce a smooth and reasonable morph when occlusions and disocclusions occur by optimally interpreting these events in the current single-layer model. By enforcing a one-to-one mapping, occlusions are usually interpreted as object size reduction and disocclusions as expansion.

Acknowledgment: This work was supported by National Science Foundation CCR-0132970.

References

ARAD, N., AND REISFELD, D. 1995. Image warping using few anchor points and radial functions. *Computer Graphics Forum 14*, 1, 35–46.

BAJCSY, R., AND KOVACIC, S. 1989. Multiresolution elastic matching. *Computer Vision, Graphics, and Image Processing 46*, 1–21.

BEAUCHEMIN, S., AND BARRON, J. 1995. The computation of optical flow. *ACM Computing Surveys 27*, 3, 433–467.

BEIER, T., AND NEELY, S. 1992. Feature-based image metamorphosis. In *SIGGRAPH 1992 Proceedings*, 35–42.

BOVIK, A. 2000. *Image and Video Processing*. Academic Press.

BROWN, L. 1992. A survey of image registration techniques. *ACM Computing Surveys 24*, 325–376.

BURT, P., AND ADELSON, E. 1983. The laplacian pyramid as a compact image code. *IEEE Trans. Communications COM-31*, 4, 532–540.

CANNY, J. 1986. A computational approach to edge detection. *IEEE Trans. Pat. Anal. Mach. Intell. 8*, 6, 679–698.

CHUANG, Y.-Y., AGARWALA, A., CURLESS, B., SALESIN, D., AND SZELISKI, R. 2002. Video matting of complex scenes. In *Proceedings of Siggraph*, 243–248.

GAO, P., AND SEDERBERG, T. 1998. A work minimization approach to image morphing. *The Visual Computer 14*, 390–400.

GOMES, J., BEIER, T., COSTA, B., DARSA, L., AND VELHO, L. 1997. Warping and morphing of graphical objects. SIGGRAPH 97 course notes.

HEEGER, D., AND BERGEN, J. 1995. Pyramid-based texture analysis/synthesis. In *Proc. of SIGGRAPH*, 229–238.

LEE, S.-Y., CHWA, K.-Y., HAHN, J., AND SHIN, S. 1995. Image morphing using deformation techniques. *Visualization and Computer Animation 6*, 3.

LEE, S.-Y., CHWA, K.-Y., SHIN, S., AND WOLBERG, G. 1995. Image metamorphosis using snakes and free-form deformations. In *SIGGRAPH 1995 Proceedings*, 439–448.

LEE, S., WOLBERG, G., AND SHIN, S. 1998. Polymorph: Morphing among multiple images. *IEEE CG&A 18*, 1.

LITWINOWICZ, P., AND WILLIAMS, L. 1994. Animating images with drawings. In *SIGGRAPH 1994 Proceedings*, 409–412.

LUCAS, B., AND KANADE, T. 1981. An iterative image registration technique with an application to stereo vision. In *Proc. 7th Intl. Joint Conf. on Art. Intell.*

MORPHMAN, 2003. video-to-video morphing. http://www.stoik.com/products/morphman/mm30_v2v.htm.

OLSON, C. 2002. Maximum-likelihood image matching. *IEEE Trans. Patt. Anal. Mach. Intell. 24*, 6, 853–857.

QUENOT, G., PAKLEZA, J., AND KOWALEWSKI, T. 1998. Particle image velocimetry with optical flow. *Experiments in Fluids 25*, 177–189.

RUPRECHT, D., AND MULLER, H. 1995. Image warping with scattered data interpolation. *IEEE Computer Graphics & Applications 15*, 1, 37–43.

SHI, J., AND TOMASI, C. 1994. Good features to track. In *IEEE Conf. Computer Vision and Pattern Recognition*.

SHINAGAWA, Y., AND KUNII, T. 1998. Unconstrained automatic image matching using multiresolutional critical-point filters. *IEEE Trans. Patt. Anal. Mach. Intell. 20*, 9, 994–1010.

SZEWCZYK, R., FERENCZ, A., ANDREWS, H., AND SMITH, B. 1997. Motion and feature-based video metamorphosis. In *ACM Multimedia*, 273–281.

TORRESANI, L., YANG, D., ALEXANDER, G., AND BREGLER, C. 2001. Tracking and modeling mon-rigid objects with rank constraints. In *IEEE Conf. Computer Vision and Pattern Recognition*.

WOLBERG, G. 1990. *Digital Image Warping*. IEEE Computer Society Press.

WOLBERG, G. 1998. Image morphing survey. *The Visual Computer 14*, 360–372.