Video Matching

Peter Sand and Seth Teller*
MIT Computer Science and Artificial Intelligence Laboratory

Abstract

This paper describes a method for bringing two videos (recorded at different times) into spatiotemporal alignment, then comparing and combining corresponding pixels for applications such as background subtraction, compositing, and increasing dynamic range. We align a pair of videos by searching for frames that best match according to a robust image registration process. This process uses locally weighted regression to interpolate and extrapolate high-likelihood image correspondences, allowing new correspondences to be discovered and refined. Image regions that cannot be matched are detected and ignored, providing robustness to changes in scene content and lighting, which allows a variety of new applications.

CR Categories: I.4.3 [Image Processing and Computer Vision]: Enhancement—Registration; I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Motion

Keywords: video alignment, robust image matching, robust image registration, high dynamic range video

1 Introduction

Given multiple still images of a scene from the same camera center, one can perform a variety of image analysis and synthesis tasks, such as foreground/background segmentation, copying an object or person from one image to another, building mosaics of the scene, and constructing high dynamic range composites.

Our goal is to extend these techniques to video footage acquired with a moving camera. Given two video sequences (recorded at separate times), we seek to spatially and temporally align the frames such that subsequent image processing can be performed on the aligned images. We assume that the input videos follow nearly identical trajectories through space, but we allow them to have different timing. The output of our algorithm is a new sequence in which each "frame" consists of a pair of registered images. The algorithm provides an alternative to the expensive and cumbersome robotic motion control systems that would normally be used to ensure registration of multiple video sequences.

The primary difficulty in this task is matching images that have substantially different appearances (Figure 1). Video sequences of the same scene may differ from one another due to moving people, changes in lighting, and/or different exposure settings. In order to obtain good alignment, our algorithm must make use of as much image information as possible, without being misled by image regions that match poorly.

Traditional methods for aligning images include feature matching and optical flow. Feature matching algorithms find a pairing of feature points from one image to another, but they do not give a dense pixel correspondence. Optical flow produces a dense pixel correspondence, but is not robust to objects present in one image but not the other.

Our method combines elements of feature matching and optical flow. In a given image, the algorithm identifies a set of textured image patches to be matched with patches in the other image. Once a set of initial matches has been found, we use these matches as motion evidence for a regression model that estimates dense pixel correspondences across the entire image. These estimates allow further matches to be discovered and refined using local optical flow. Throughout the process, we estimate and utilize probabilistic weights for each correspondence, allowing the algorithm to detect and discard (or fix) mismatches.

Our primary contribution is a method for spatially and temporally aligning videos using image comparisons. Our image comparison method is also novel, insofar as it is explicitly designed to handle large-scale differences between the images. The main limitation of our approach is that we require the input videos to follow spatially similar camera trajectories. The algorithm cannot align images from substantially different viewpoints, partially because it does not model occlusion boundaries. Nonetheless, we demonstrate a variety of applications for which our method is useful.

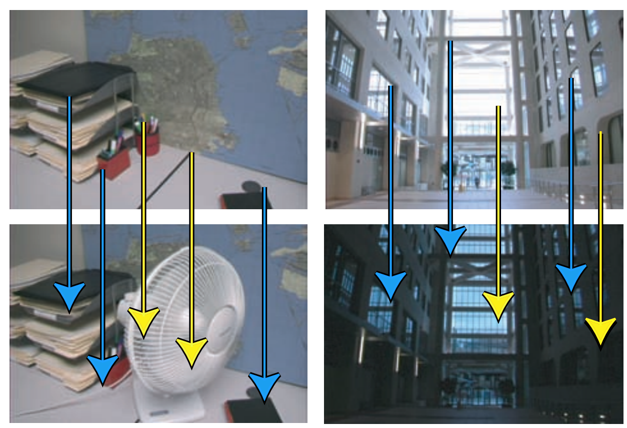

Figure 1: Our matching algorithm is robust to differences such as an object that appears in only one image (left pair) or changes in lighting and exposure (right pair). The key idea behind our matching algorithm is to identify which parts of the image can be matched (blue arrows) without being confused by parts of the image that are difficult or impossible to match (yellow arrows).

2 Related Work

Aligning a pair of images is a standard problem in computer vision. Optical flow algorithms [Beauchemin and Barron 1995] find a vector field that maps each pixel from one image to a corresponding pixel in another image. Stereo methods [Scharstein and Szeliski 2002] use known camera poses to restrict the search to 1D lines (for images of a static scene). Many of these algorithms are robust to small-scale effects (such as local violations of smoothness

*e-mail: {sand,teller}@csail.mit.edu

or reflectance assumptions), but they are not intended for matching images that have large differences in lighting or have large objects that appear in one image but not the other. Some flow estimation methods [Black and Anandan 1996] handle large image regions that do not match by robustly fitting global parametric models to local flow estimates. Our method fits a non-parametric model and does not rely entirely on local flow estimates.

The basis of our algorithm is matching salient image points [Harris and Stephens 1988; Brown and Lowe 2003]. Many existing methods prune feature matches using robust fitting methods (such as RANSAC [Fischler and Bolles 1981]) with constraints from the fundamental matrix [Hartley and Zisserman 2000]. Brown and Lowe [2003] align images by matching features that are invariant to several spatial and illumination transformations. Kanazawa and Kanatani [2002] find good correspondences using epipolar constraints combined with smoothness and spatial consistency criteria. Smith *et al.* [1998] refine feature matches by comparing the length and angle of each correspondence vector with its neighbors.

None of these image correspondence techniques addresses the larger problem of video registration. Caspi and Irani [2000] align video sequences using a single image transformation and single time offset for an entire sequence. This method is successful for rigidly connected cameras that simultaneously record a scene, but does not apply to spatially or temporally different motions. Sawhney *et al.* [2001] provide a method of aligning two video sequences using stereo and optical flow, but also aim only at the case of rigidly connected cameras simultaneously recording a scene. Rao *et al.* [2003] temporally match video sequences by tracking a feature that appears in each video and aligning the resulting trajectories. This requires a user-specified trackable feature and does not provide dense pixel correspondences between the video sequences.

Our method provides a warping field and temporal offset for each frame, allowing the video frames to be registered for various segmentation and compositing applications. Several of these applications have been addressed via different methods. Chuang *et al.* [2002] use mosaicing techniques to reconstruct a background image that is used for foreground segmentation. Kang *et al.* [2003] register images at different exposures to obtain high dynamic range video. These applications and others can be performed with the help of the method we present in this paper.

3 Overview

Our goal is to construct a mapping between two videos so that both videos can be manipulated in a shared spatial and temporal domain. One of the two videos is designated as the primary video, the other as the secondary. The primary video provides the spatial and temporal reference; the secondary video is mapped to match it.

The core of the algorithm is robust image alignment, described in Section 4, which provides a warping from one image to another that is robust to significant differences between those images. This image alignment technique is used as a sub-function of the video alignment process, which is described in Section 5. In Section 6, we present two extensions to the basic algorithm. We describe experimental evaluations in Section 7 and give various applications in Section 8. Limitations and planned solutions to these limitations are discussed in Section 9.

4 Robust Image Alignment

Our image alignment algorithm finds correspondences between pixels in a pair of images. Each correspondence is assigned a weight according to the likelihood that it describes a physical 3D point undergoing a physical 3D motion. The ability to characterize the correctness of a correspondence is essential to the robustness of the algorithm. We want to use as much information from the images as possible, but we do not want to be misled by unexpected differences between the images.

The weight w_i assigned to the i^{th}correspondence is the product of two terms: a pixel matching probability P_i (Section 4.1) and a motion consistency probability M_i (Section 4.2). For simplicity, we assume independence when combining the probabilities.

4.1 Pixel Consistency

To compute the pixel matching probability, P_i, for a particular correspondence, we evaluate how well the images match in a square region around the correspondence. Rather than simply comparing pixel values, we use a method that allows small spatial variations in the corresponding pixel locations. This technique, inspired by Birchfield and Tomasi [1998], permits small changes in scale, rotation, and skew of an image region due to differences in camera viewpoint. This also alleviates several sampling issues. (Similar methods are proposed by Kutulakos [2000] and Szeliski and Scharstein [2002].)

A single pixel in the primary image is compared with a 3-by-3 neighborhood of pixels in the secondary image, rather than with a single secondary pixel. To do this efficiently, the algorithm applies 3-by-3 minimum and maximum filters to the secondary image, producing new images I_{min} and I_{max} (Figure 2). These minimum and maximum images define bounds on the value of each pixel in the secondary image; the corresponding primary pixel receives a penalty if and only if its value lies outside this interval.

To evaluate a correspondence, our algorithm sums this pixel matching score across a square region (with size specified in Section 7). For an image region R in the primary image I we obtain the following score:

$$\sum_{(x,y)\in R} max(0, I(x,y) - I_{max}(x+u,y+v), I_{min}(x+u,y+v) - I(x,y)). \tag{1}$$

Here u and v describe an offset from a point (x,y) in the primary image to the corresponding point $(x+u,y+v)$ in the secondary image (the same offset is used across the entire region). We average the above score over each color channel to obtain the pixel intensity dissimilarity d_i for the i^{th}correspondence.

In the case that the primary and secondary images contain substantial differences in lighting or exposure, we perform local brightness and contrast normalization [Sand and Teller 2004], then use the same min/max image comparison on the normalized images.

In either case, we use the pixel intensity dissimilarity d_i to compute the pixel matching probability P_i:

$$P_i = N(d_i, \sigma^2_{pixel}). \tag{2}$$

Here $N(x,\sigma^2)$ is a zero-mean Gaussian with variance σ^2 evaluated at x. We specify σ_{pixel} as described in Section 7.

This method of comparing image regions attains some invariance to affine transformations, but not as much as other methods [Brown and Lowe 2003; Ferrari et al. 2001]. Strong invariance is not necessary for our algorithm, because we limit the input images to have similar viewpoints.

4.2 Motion Regression and Consistency

To evaluate motion consistency, we determine how well the offset vector (u,v) of a particular correspondence agrees with its neighbors. This requires initial estimates of the weights $\{w_i\}$ for the other correspondences, which we will obtain as described in Section 4.3.

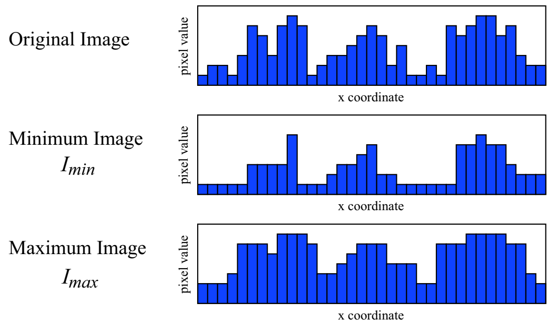

Figure 2: Each plot represents a cross section of a hypothetical image. The image is (non-linearly) filtered so each pixel becomes the minimum or maximum of its 3 by 3 neighborhood.

From these weights and the correspondences $\{(x_i, y_i, u_i, v_i)\}$, the algorithm reconstructs a vector field $u(x,y)$, $v(x,y)$ that provides an offset for each pixel of the primary image.

Our algorithm computes $u(x,y)$ and $v(x,y)$ using locally weighted linear regression [Atkeson et al. 1997], which determines the value of a function at a query point by fitting a regression model to nearby points, each weighted by its distance to the query point. The smoothness of locally weighted regression is determined by a kernel width parameter, K, describing the shape of the distance weighting function (typically a Gaussian). We modify this method to incorporate our correspondence probabilities by multiplying the kernel weight for each correspondence by its matching weight w_i.

We make one additional modification to standard locally weighted regression: we adapt the kernel width according to the density of points around the query point (Figure 3). We increase the kernel width K in regions of low data density (to bridge large gaps) and decrease K in regions of high data density (to model fine details). To do this, the algorithm sets K to the average distance from the query point to the N nearest neighbors. (N is one of the parameter values given in Section 7.) This adaptive kernel width is particularly useful for image correspondences, which may occur densely in highly texture regions, but very sparsely elsewhere (such as untextured walls and floors).

A linear model for u and v in terms of x and y can describe image-space rotation, scaling, and other affine transformations. By using locally weighted regression, we extend the linear model to describe smooth image warps, including lens distortion and gradual variations due to depth and perspective. One advantage of fitting a local model is that we expect to extrapolate better than simply averaging nearby points (Figure 3).

In order to compute the motion consistency probability M_i for a correspondence, the algorithm compares the previously assigned vector (u_i, v_i) with the vector $(\hat{u}_i, \hat{v}_i)$ predicted by adaptive locally weighted regression. The motion consistency probability is based on the difference between these two vectors:

$$M_i = N\left(\sqrt{(u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2}, \sigma^2_{motion}\right). \quad (3)$$

We experimented with a fundamental matrix model but found that it was redundant with the motion regression; in our test sets, the correspondences that satisfy the fundamental matrix also have high motion consistency probability.

4.3 Finding Good Correspondences

Now that we have a way of evaluating the quality of a correspondence, we can attempt to find a number of good correspondences between a pair of images. To compute the motion consistency probabilities, we must bootstrap the algorithm with some good initial guesses.

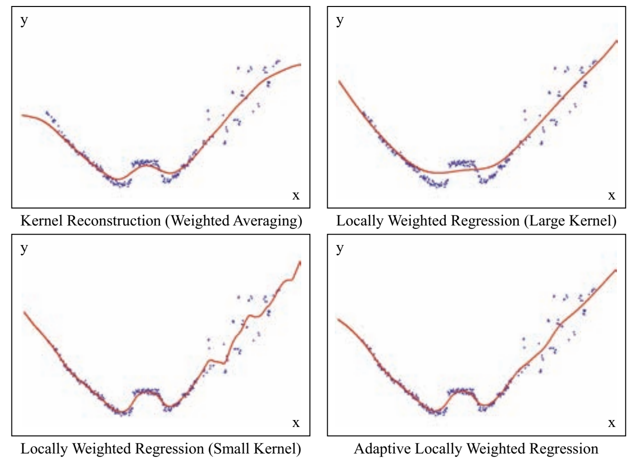

Figure 3: Each plot represents a generic regression problem in which we seek to fit a function $y(x)$. Weighted averaging does not extrapolate the function beyond the given data. Locally weighted linear regression does extrapolate, but leaves an issue of selecting the best kernel size. When the data density is highly variable, we prefer to adjust the kernel size according to the local density. We use adaptive locally weighted regression to interpolate and extrapolate correspondences, resulting in a dense correspondence field.

The algorithm begins by selecting feature points using a Harris corner detector [Harris and Stephens 1988] (with a modification from page 45 of Noble's thesis [1989]). Each feature point in the primary image is compared with the feature points in the secondary image to find good matches according to nearby pixel values. These initial matches are used to find preliminary regression predictions.

For each feature point in the primary image, we then search for the most likely match in the secondary image according to the correspondence weighting function (including both pixel matching and motion consistency). The algorithm checks for matches in the secondary image at the location predicted by the regression function and at various nearby feature points found by the corner detector. For each candidate location, the algorithm performs a local motion optimization using the KLT method [Lucas and Kanade 1981; Shi and Tomasi 1994]. Because the KLT optimizer is initialized with regression predictions, it can find good correspondences even when the feature detector fails to find the same points in each image. The local motion optimization allows sub-pixel correspondences, which we would not obtain simply by matching feature-detector maxima.

After trying to improve each correspondence, the algorithm recomputes the regression predictions and repeats the pointwise correspondence optimization (in a manner similar to EM [Dempster et al. 1977]). Termination occurs when an iteration completes without making further improvement.

An advantage of this EM-like method over an alternative such as RANSAC [Fischler and Bolles 1981] is that our algorithm can alter the correspondences (through the use of regression and local motion optimization) to obtain better correspondences after an initial pairing. In an image matching context (as opposed to 3D reconstruction), Kanazawa and Kanatani [2002] demonstrate that an iterative feedback algorithm performs better than RANSAC.

After finding a set of high likelihood correspondences, we use the locally weighted regression method described in Section 4.2 to interpolate and extrapolate the offset vectors, obtaining a dense correspondence field.

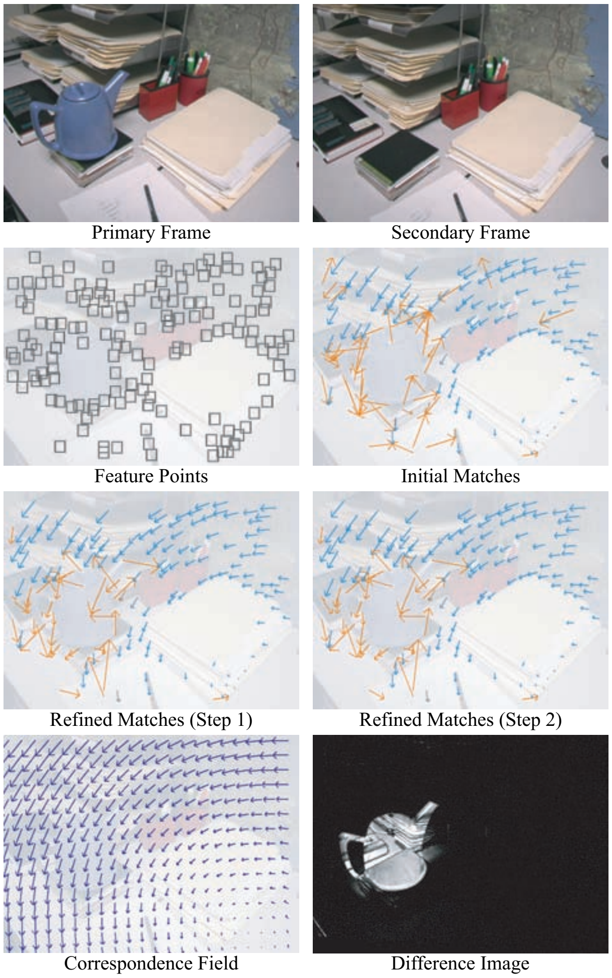

Figure 4: The image matching algorithm typically converges in a few iterations. The blue and yellow arrows denote high and low probability correspondences, respectively. The algorithm successfully recognizes that the teapot pixels cannot be matched with the background. The reconstructed dense correspondences are quite accurate, as illustrated by the difference between the primary frame and the warped secondary frame. The black regions in the difference image indicate that the background pixels are successfully matched (with a pixel difference near zero).

5 Video Matching

The robust image alignment method described in the previous section is the primary component of our video matching algorithm. Given the image alignment method, the video matching process is relatively simple. We search for possible pairings between frames in the primary and secondary videos using the image alignment algorithm to evaluate candidate frame matches.

For each primary frame, once a matching secondary frame has been found, the secondary frame is warped into alignment with the primary frame. The output of the algorithm is a new version of the secondary video that is spatially and temporally registered with the primary video.

5.1 Frame Matching Measure

To evaluate the quality of a match between a pair of frames, we use the robust image alignment method (Section 4) to find a correspondence field $u(x,y)$, $v(x,y)$ between the frames, then use it to estimate how well the primary and secondary frames match.

Our frame matching objective function has two parts: a parallax measure and the correspondence vector magnitude. We minimize parallax because depth discontinuities will cause errors in the reconstructed correspondence field. Correspondence magnitude is less important, but we nonetheless minimize it to obtain maximal overlap between the frames. To evaluate the match between frames i and j, we take a weighted combination of the parallax quantity $p_{i,j}$ and the mean correspondence vector magnitude $m_{i,j}$:

$$D_{i,j} = \lambda \cdot p_{i,j}^2 + m_{i,j}^2. \tag{4}$$

Our parallax measure $p_{i,j}$ quantifies the amount of depth-induced relative motion between the correspondences. Given a pair of correspondences, we compute the distance between the points in the primary image and the distance between the points in the secondary image. We define $p_{i,j}$ to be the change in this pairwise distance between the primary and secondary images, averaged over all pairs of correspondences. This measure is invariant to image-plane rotation and translation, but not invariant to looming motions and depth effects (the quantities we wish to detect).

We square each term so that the objective function is essentially quadratic for linear motions (to ease optimization). We set λ to 5 to capture the relative importance of parallax over magnitude.

5.2 Adaptive Search for Matching Frames

Using the objective function described in Section 5.1, we wish to search the secondary video for a good match to a particular frame in the primary video. For computational efficiency, we do not want to evaluate the objective function for every frame of the secondary video, but instead select a small subset of frames to consider.

Given some initial guess of where to look in the secondary video, our algorithm evaluates several nearby frames and fits a quadratic regression model to the objective function values of these pairings (Figure 5). These preliminary evaluations occur at the initial guess, 1 frame forward, 1 frame backward, 5 frames forward, and 5 frames backward. The algorithm then checks frames near the minimum of the quadratic model and re-estimates the model after each new observation of the objective function. Once all secondary frames near the quadratic minimum have been checked, the algorithm picks the one with the lowest objective function value.

In order to compute an initial guess for the next frame search, the algorithm computes a weighted average of the changes between the frame indices of the prior matches. The weights decay by $1/2$ for each frame and are truncated after 5 frames. This weighted average is added to the previous frame index to obtain a starting point for the search for the next frame. The decaying weights allow the algorithm to respond to changes in the relative camera velocity between the videos, but with some damping to avoid over-reacting to these changes.

For the first frame of the primary video, we have no previous evidence for where to look in the secondary video. We do not need to know the particular frame that will match best, but we need a good enough guess to initiate the quadratic search. This initial guess can be provided by the user or found automatically by a linear search of the secondary video.

This search method allows substantial flexibility in the temporal mapping from one video to the other. One video can be much faster than the other or proceed in the opposite direction. The videos can change speed and relative direction, so long as the changes are smooth. A video graph (Section 6) can be used to handle discontinuous temporal mappings.

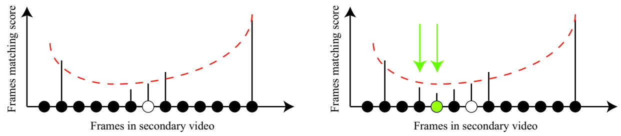

Figure 5: Given an initial guess (white circle) of which frame to use in the secondary video, we check several nearby frames (left). We fit a quadratic regression model to the frame matching scores (red dashed curve), then check frames near the minimum of the curve (green arrows). Next we refit the quadratic model and repeat the process until all near-minimal frames have been checked. Finally, we pick the frame with the lowest score (green circle).

6 Optional Variations

Fast frame matching. In order to speed up the video matching process, we quickly estimate the quality of the match between a pair of frames. To do this, we run the image matching algorithm (Section 4.3), but skip the KLT motion optimization (the part of the process that takes the most time). This results in less accurate correspondences, but does not substantially affect the correspondence properties that are used to select matching frames as described in Section 5.1. Once we have found a good frame match, we re-run the full algorithm to obtain accurate pixel correspondences.

Video graph matching. For some applications, the secondary video may include many passes over a single background environment. In this case, rather than searching for frame matches within a temporal window of the second video, we would like to consider possible matches scattered throughout the video. To do this, we build a video graph, in which each frame is a node and edges are created between frames that have a similar pose, as determined by the image alignment algorithm. (A video graph is like a video texture [Schödl et al. 2000], but designed to handle variations in camera pose.) To perform video matching, the algorithm searches the graph for good matches, starting at the best match found for the previous frame. Thus the search considers a range of nearby viewpoints, regardless of their original temporal ordering. Additional details are available in the technical report [Sand and Teller 2004].

7 Experimental Evaluation

To characterize the quality of a video match, we warp each secondary frame into the corresponding primary frame and compare pixel values. To avoid sampling artifacts, we use the min/max pixel comparison method described in Section 4.1. We take the mean over the pixels in each frame (not including the pixels for which the primary and secondary frames do not overlap), averaged over all the frames in the primary video sequence.

This produces a single number that represents the quality of the video match. Using this measure, we can explore various design changes (such as verifying that fundamental matrix constraints do not improve the results). We can also set the algorithm's parameters by determining which values give the best scores on a training set.

For our experiments, we set the algorithm's parameters using a training set consisting of a variety of different sequences (with different kinds of motion and different kinds of scenes). Because there is little danger of overfitting, we expect these same parameters to perform well on other sequences. For the feature detector (Section 4.3), we use a Gaussian window with a standard deviation of 5 pixels and a detector threshold of 1.0. The feature detector enforces a minimum spacing of 12 pixels between feature points. The algorithm computes the pixel dissimilarity for a correspondence using a 24 by 24 pixel region. The search for initial matches is restricted to be within 100 pixels of each primary frame feature point. We use the average distance to the nearest 80 points to set the adaptive kernel width for locally weighted regression. We set $\sigma_{pixel} = 2$ and $\sigma_{motion} = 10$.

On a set of 200 image pairs from four different kinds of scenes, the algorithm had an average running time of 1.31 seconds for each image pair (on a single-processor desktop PC). The majority of this time is spent on the KLT optimization described in Section 4.3. Performing the complete video matching algorithm (with multiple image comparisons per frame) takes several minutes per second of primary video. The fast matching method described in Section 6 improves the overall running time by about a factor of seven.

8 Applications

The ability to register video sequences has a variety of applications. As illustrated by Agarwala *et al.* [2004], a set of registered images provides numerous opportunities for image manipulation. The video matching algorithm described in this paper allows these operations to be performed on frame sequences from moving cameras. We demonstrate several of these applications in the video available on the 2004 SIGGRAPH DVD. These demos are described in more detail in the technical report [Sand and Teller 2004].

The output of the video matching algorithm is a new version of the secondary video in which each frame is registered with the corresponding frame of the primary video. Given this aligned secondary video, pixels can be copied over, compared with, and blended with pixels from the primary video using standard commercial compositing software.

Background subtraction. Given an image containing objects and an empty background image without the objects, the objects can be localized by comparing corresponding pixels (Figure 4). Tracking a moving object enables tasks such as gesture recognition, surveillance, and markerless motion capture [Davison et al. 2001]. If the object is visually different from the background, accurate object boundaries can be found, providing mattes for various of filmmaking applications. These mattes can be improved using more sophisticated methods [Chuang et al. 2002].

Compositing. Aligning two video sequences allows pixels to be copied from one to the other (Figure 9). An empty street with an action sequence can be composited onto a street full of traffic. A blue sky can be composited onto a shot that had a gray sky. People and objects can be added to or repeated in a scene. In many cases, a rough matte is sufficient for this kind of compositing, because the background is assumed to be the same in both sequences. In some of our demos, we use the difference images to guide the motion of a rough matte. In other cases (such as causing a glass of orange juice to become empty), we key-frame the motion of the rough matte. If a precise matte is needed, it can be obtained by background subtraction, color segmentation, or semi-automatic rotoscoping.

Automatic wire removal. One particular kind of compositing that occurs frequently in special effects work is wire removal. After filming an empty background sequence, we can automatically remove wires using a mask attained via background subtraction. Through image filtering, our algorithm detects which parts of the mask occur in thin lines and copies background pixels at these locations (Figure 6). Cranes, platforms, and other rigging can be removed in a similar fashion, though approximate mattes may need to be manually specified for more complex objects.

Replacement of stand-ins. A couple recent films have used actors to stand in for CG characters in order to provide a reference for other actors and/or computer animators. These stand-in actors must be replaced with the scene background when the CG character is composited into the shot (unless the CG character happens to overlap the stand-in in every pixel of every frame). Video matching can replace some of the extensive manual labor that has been used to paint a background over stand-ins for CG characters.

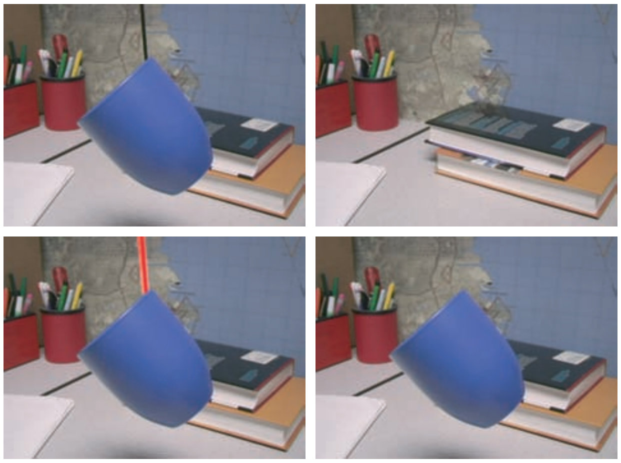

Figure 6: The top images have been registered using the robust matching algorithm. From these images we can use simple image processing methods (background subtraction and color thresholds) to create a mask for the wire (in red). Inside the mask, we copy pixels from the background. This allows a wire to be automatically removed in each frame of a long sequence from a moving camera.

Wide field-of-view video. By matching an overlapping part of two video sequences, our method can merge them into one video sequence with a larger field of view. This differs from prior mosaicing methods [Brown and Lowe 2003] insofar as we produce a separate mosaic for each frame. Multiple secondary videos can expand the per-frame mosaics, so long as each sequence overlaps with another. One limitation of this approach is that moving objects cannot move from one sequence to another, unless the sequences are recorded at the same time (by placing multiple cameras on a rig).

High dynamic range video. Attaining proper exposure is one of the most common and difficult problems in filmmaking. A particularly useful kind of compositing is the creation of high dynamic range video from several low dynamic range videos recorded at different exposures (Figure 7) [Debevec and Malik 1997]. To perform video matching across different exposures, we first normalize the local contrast and brightness (Section 4.1). In this case, we set $\sigma_{pixel} = 5$ and $\sigma_{motion} = 5$. Once the sequences have been aligned, standard techniques can be used to combine the images and remap the result for display [Debevec and Malik 1997; Kang et al. 2003]. This approach can be performed on scenes that involve a moving subject, which must be properly exposed in one sequence while other sequences (without the moving subject) properly expose various parts of the background.

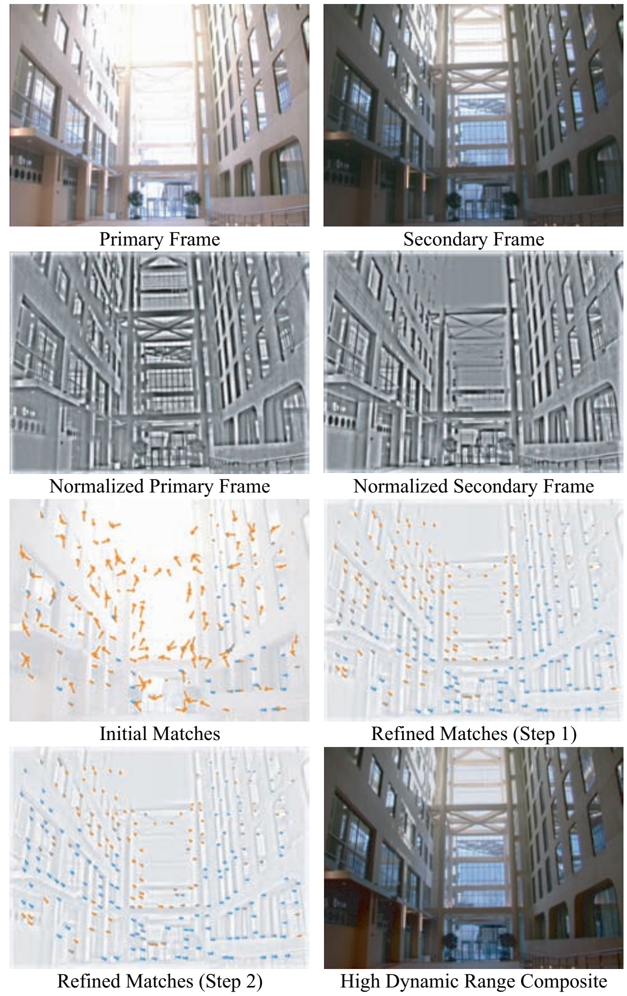

Figure 7: These frames were selected by the video matching algorithm from a pair of videos recorded at different exposures. The algorithm first performs local brightness/contrast normalization, then finds high-likelihood correspondences. Once the secondary frame has been mapped to the first, the exposures are combined to create a high dynamic range composite.

9 Limitations and Future Work

The main limitation of our approach is that the primary and secondary video sequences must have spatially similar motions. Our method allows general camera motion (hand-held, tripod-mounted, vehicle-mounted, etc.), but requires that each video sequence follow nearly the same trajectory through space (though perhaps with substantially different timing). This limitation arises partially because our algorithm does not model discontinuities in the correspondence field. We do represent variation in pixel motion due to depth, but we assume that this variation occurs smoothly across the image. Another limitation is the algorithm's dependence on 2D image texture for matching.

Both discontinuities and a lack of 2D texture are issues that are handled by many existing optical flow algorithms. However, these algorithms cannot cope with large-scale differences between the images (such as an object that appears in one image but not the other). When large image regions are unmatchable, we have barely enough information to find a smooth warping between the images; finding correct discontinuities can be difficult if not impossible.

Nonetheless, in the future we plan to extend our algorithm to make better use of the information in the images. We intend to decompose the optical motion into depth parameters and camera motion parameters. To do this, the algorithm described in this paper will be used to find correspondences for the estimation of epipolar geometry. The epipolar constraints can then be used to incorporate information from 1D edge features (not just 2D features), resulting in a better correspondence field.

We also plan to combine both inter-sequence and intra-sequence matching to improve temporal coherence and frame search efficiency. Based on the inter-sequence correspondences for one frame pair, we will select parts of each frame in which to compute intra-sequence correspondences. These intra-sequence correspondences will then be used to select a secondary frame that will match the subsequent primary frame.

10 Conclusion

This paper presents a new method for registering multiple video sequences by selecting video frames and applying image warps. We provide a method for robust image alignment that combines elements of feature-point correspondence matching and local motion estimation (i.e. optical flow). Unlike existing methods, the algorithm is explicitly designed to handle large-scale differences between images. Our method makes effective use of the information available in the image without being distracted by parts of the image that are not matchable.

We use this image registration method as a sub-routine in a video alignment algorithm that searches for a good match for each video frame. This algorithm provides a partial solution to the problem of aligning video sequences that were recorded with general camera motions. This is a valuable problem to solve and one that has attracted relatively little attention in the past.

As discussed in Section 9, the main limitations of this method are that the videos must follow spatially similar camera trajectories and that the videos must contain sufficient texture. Both of these limitations can be partially overcome by incorporating 1D image constraints.

Despite these limitations, the algorithm is useful for a variety of applications, such as foreground segmentation, compositing, wire removal, replacing stand-ins, per-frame mosaicing, and high dynamic range imaging. In the past, many of these applications required registered images from a static or robotically controlled camera. These techniques can now be applied in a wider range of situations using the methods presented in this paper.

References

AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUCKER, S., COLBURN, A., CURLESS, B., SALESIN, D., AND COHEN, M. 2004. Interactive digital photomontage. *ACM Trans. Graph.*, In press.

ATKESON, C. G., MOORE, A. W., AND SCHAAL, S. 1997. Locally weighted learning. *Artificial Intelligence Review 11*, 1-5, 11–73.

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

BIRCHFIELD, S., AND TOMASI, C. 1998. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Trans. on Pattern Analysis and Machine Intelligence 20*, 4, 401–406.

BLACK, M. J., AND ANANDAN, P. 1996. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding 63*, 1, 75–104.

BROWN, M., AND LOWE, D. G. 2003. Recognising panoramas. In *ICCV*, 1218–1225.

CASPI, Y., AND IRANI, M. 2000. A step towards sequence to sequence alignment. In *CVPR*, 682–689.

CHUANG, Y.-Y., AGARWALA, A., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2002. Video matting of complex scenes. *ACM Trans. Graph. 21*, 3, 243–248.

DAVISON, A. J., DEUTSCHER, J., AND REID, I. D. 2001. Markerless motion capture of complex full-body movement for character animation. In *Eurographics Workshop on Animation and Simulation*, 3–14.

DEBEVEC, P. E., AND MALIK, J. 1997. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH*, 369–378.

DEMPSTER, A. P., LAIRD, N. M., AND RUBIN, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B 39*, 1, 1–38.

FERRARI, V., TUYTELAARS, T., AND VAN GOOL, L. 2001. Real-time affine region tracking and coplanar grouping. In *CVPR*, 226–233.

FISCHLER, M. A., AND BOLLES, R. C. 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.

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

HARTLEY, R., AND ZISSERMAN, A. 2000. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK.

KANAZAWA, Y., AND KANATANI, K. 2002. Robust image matching under a large disparity. In *Workshop on Science of Computer Vision*, 46–52.

KANG, S. B., UYTTENDAELE, M., WINDER, S., AND SZELISKI, R. 2003. High dynamic range video. *ACM Trans. Graph. 22*, 3, 319–325.

KUTULAKOS, K. N. 2000. Approximate N-view stereo. In *ECCV*, 67–83.

LUCAS, B., AND KANADE, T. 1981. An iterative image registration technique with an application to stereo vision. In *Int. Joint Conf. Artificial Intelligence*, 674–679.

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

RAO, C., GRITAI, A., AND SHAH, M. 2003. View-invariant alignment and matching of video sequences. In *ICCV*, 939–945.

SAND, P., AND TELLER, S. 2004. Video matching. Tech. Rep. LCS TR 947, MIT.

SAWHNEY, H. S., 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 *SIGGRAPH*, 451–460.

SCHARSTEIN, D., AND SZELISKI, R. 2002. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision 47*, 1-3, 7–42.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. In *SIGGRAPH*, 489–498.

SHI, J., AND TOMASI, C. 1994. Good features to track. In *CVPR*, 593–600.

SMITH, P., SINCLAIR, D., CIPOLLA, R., AND WOOD, K. 1998. Effective corner matching. In *British Machine Vision Conference*, 545–556.

SZELISKI, R., AND SCHARSTEIN, D. 2002. Symmetric sub-pixel stereo matching. In *ECCV*, 525–540.

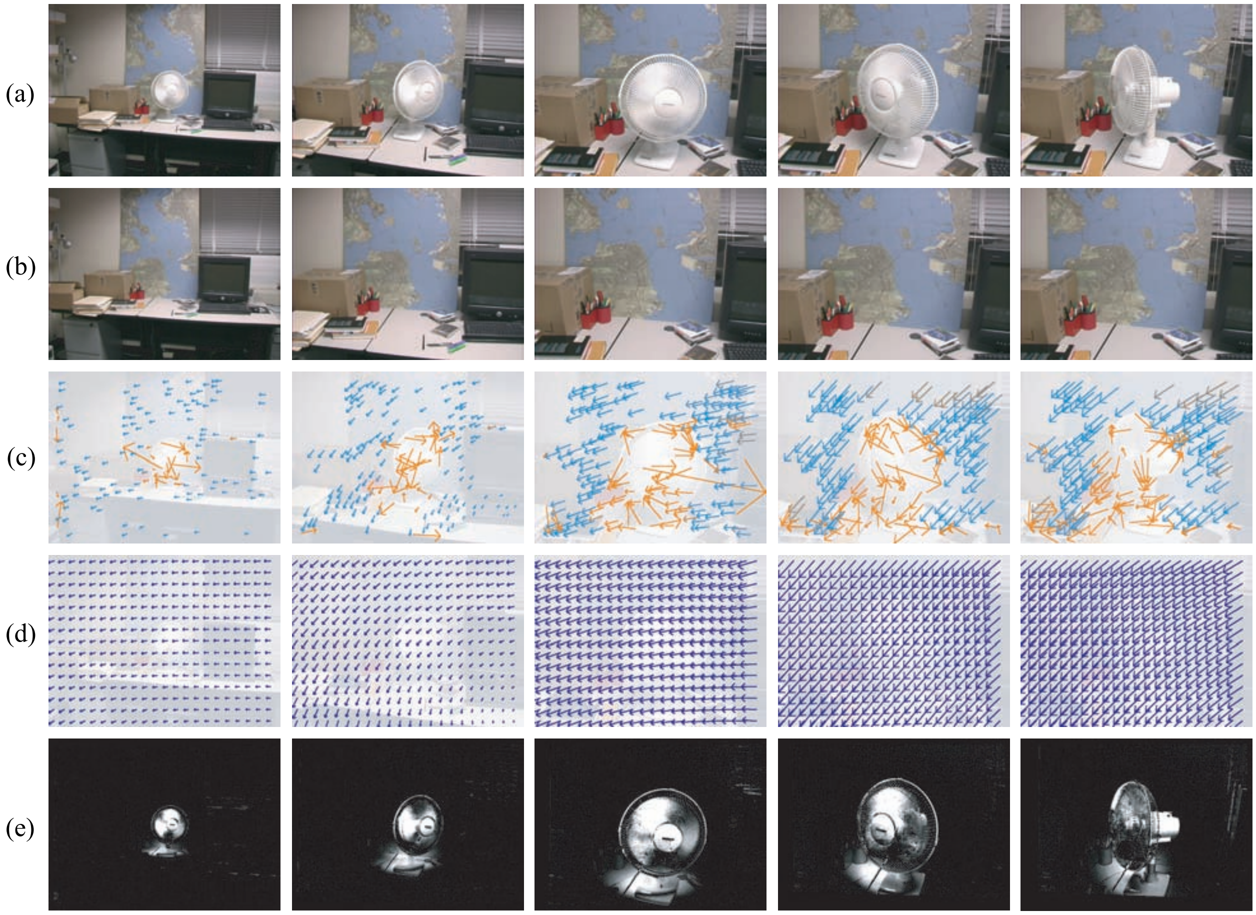

Figure 8: **(a)** Primary video frames from a hand-held sequence (frames 0, 60, 120, 180, 240). **(b)** Matching secondary video frames found by our algorithm (frames 14, 89, 138, 147, 147). **(c)** Refined correspondences found by the algorithm. **(d)** Reconstructed correspondence fields. **(e)** Difference between primary frame and projected secondary frame.

Figure 9: This figure shows primary frames **(a)**, secondary frames **(b)**, and various composites **(c)**. From left to right: **(1)** a transparent fan created by blending the two frames, **(2)** color manipulated according to a difference matte, **(3)** cloning a person by compositing the left half of one image with right half of another, **(4)** changing the amount of orange juice using a horizontal compositing line, and **(5)** a dismembered hand with a key-framed compositing line. None of the composites require per-frame manual rotoscoping.