Video Diff: Highlighting Differences Between Similar Actions in Videos

Guha Balakrishnan, Frédo Durand, John Guttag[*]
MIT

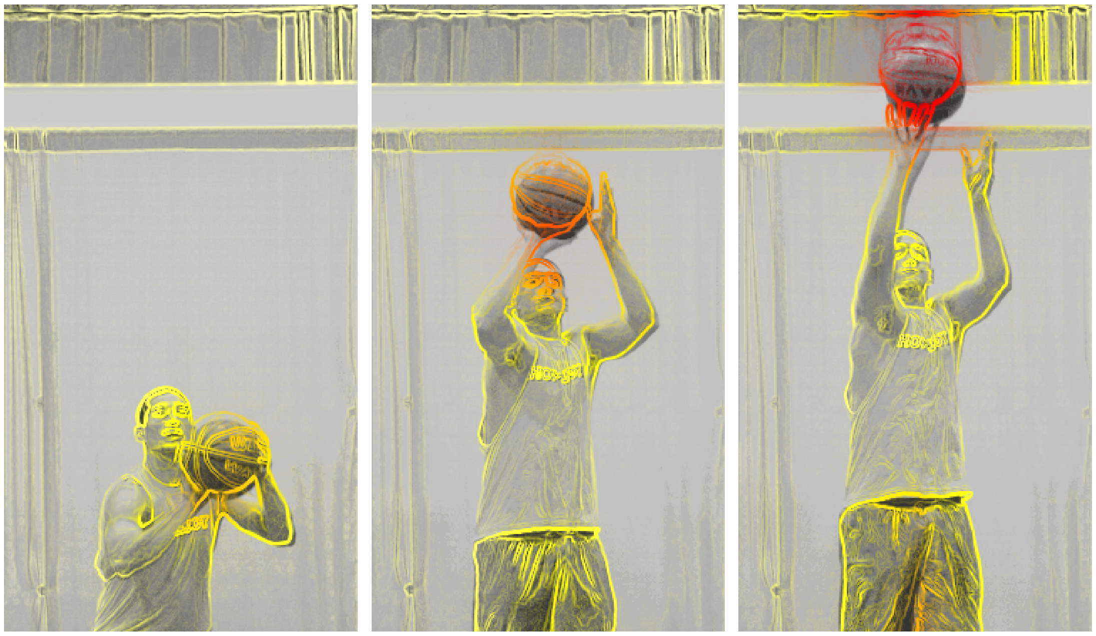

Figure 1: *Example of output frames of our method for a basketball sequence. We overlay the edges of one video over the frames of the second video. The edges are colored to signify regions of dissimilarity (red being most dissimilar, yellow being least). The overlay is meant to help a user quickly identify differences in motion between pairs of video that look very similar*

Abstract

When looking at videos of very similar actions with the naked eye, it is often difficult to notice subtle motion differences between them. In this paper we introduce video diffing, an algorithm that highlights the important differences between a pair of video recordings of similar actions. We overlay the edges of one video onto the frames of the second, and color the edges based on a measure of local dissimilarity between the videos. We measure dissimilarity by extracting spatiotemporal gradients from both videos and calculating how dissimilar histograms of these gradients are at varying spatial scales. We performed a user study with 54 people to compare the ease with which users could use our method to find differences. Users gave our method an average grade of 4.04 out of 5 for ease of use, compared to 3.48 and 2.08 for two baseline approaches. Anecdotal results also show that our overlays are useful in the specific use cases of professional golf instruction and analysis of animal locomotion simulations.

[*]e-mail:balakg@mit.edu, fredo@mit.edu, guttag@mit.edu

CR Categories: I.4.8 [Computer Vision]: Scene Analysis—Motion; I.4.7 [Computer Vision]: Feature Measurement—Feature representation; I.4.9 [Computer Vision]: Applications—; I.3.8 [Graphics]: Applications—;

Keywords: video, motion, visualization, spatiotemporal gradients

1 Introduction

Detecting subtle differences in motions is useful in such diverse areas as physical therapy, stroke recovery, machinery testing and sports. Unfortunately, when looking at videos of very similar motions with the naked eye, it is often difficult to notice subtle differences between them. In this paper we present a computational videography technique to help people visualize differences between two videos. We emphasize the application of our technique to sports, where it can be used both as an instructional tool and as a method of enhancing the viewing experience of fans. However, it can be applied to many other domains as well.

The obvious way to compare a pair of videos is to view them side by side (Fig. 2a). This is what is typically done with existing commercial sport video analysis software such as Ubersense [2015]. But detecting subtle spatial differences this way is difficult because the user must scan back and forth between the videos. Another approach is to create one video by combining the frames of the input videos, e.g., by assigning each input to separate color channels (Fig. 2b). This is the technique used in Matlab's "imfuse" function for compositing images. But this often leads to superimpositions that are hard to interpret, and require the input videos themselves

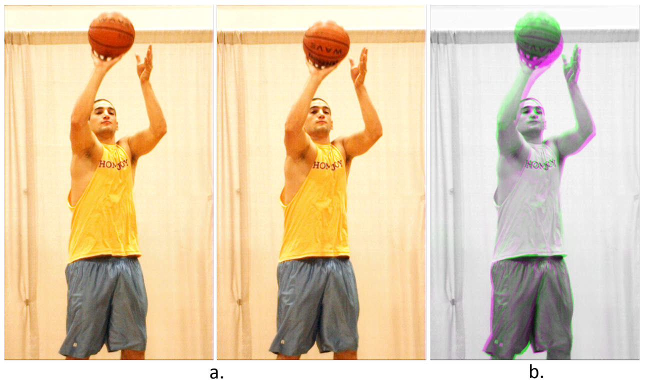

Figure 2: *Approaches to comparing very similar videos. (a.) shows frames of each video side-by-side. This is not conducive to finding differences easily. (b) shows a false-color channel overlay, where one video is mapped to the red and blue channels and the other is mapped to the green channel. Areas of magenta and green indicate differences. But this approach can be hard to interpret even when referring to the input videos.*

for reference. Our work presents a "video diffing" algorithm that takes in two videos as input and produces a single video that makes it relatively easy to spot important differences (Fig. 1). Like diffing tools for text files, our method merges two inputs into one output and indicates areas of differences.

The first challenge is detecting potentially relevant dissimilarities between videos even when the foregrounds are not the same. The second challenge is developing a way to present the information in a useful manner. We cannot use optical flow to address the first challenge because the scenes may not be identical. Instead, we first roughly align the videos in space and time and then measure local dissimilarities using a pyramid of histograms of spatiotemporal gradients. The histogram features provide illumination invariance which allows us to compare foregrounds of different appearance. We address the second challenge by overlaying the edges of one video over the frames of the second video (Fig. 1), an approach that is more interpretable than simply merging the frames as in Fig. 2b. We also color the edges based on the level of local dissimilarity between the input videos. These colored edges are meant to draw the user's attention to differences between the two videos.

Our method is simple, but dramatically enhances one's ability to notice differences. Humans are better at interpreting spatiotemporal matches than current computer vision algorithms, so instead of automatically matching video pixels, we choose to direct users to important areas of the videos, and let them interpret the differences. This design decision makes our method robust to a variety of videos.

As far as we know, this is the first work focused on diffing two videos for motion analysis. Our work is inspired by the field of motion magnification [Wadhwa et al. 2013; Wu et al. 2012], but differs in significant ways. First, motion magnification is concerned with amplifying small motions in one video, while we are focused on highlighting motion differences between two videos. Second, motion magnification assumes that all motions are small. In our case, the spatial differences are of a magnitude of many pixels. Because a large fraction of our differences violate the small motion assumption of these magnification algorithms, these differences cannot be successfully magnified.

We evaluated our method with a user study involving 54 subjects, and by having a PGA-certified golf instructor use our method to evaluate the swings of a client. The results of our user study indicate that users overwhelmingly prefer our method over viewing videos side-by-side or merging videos into different color channels. The golf instructor reported being surprised at his ability to discover aspects of his client's swings that he had not detected either by watching the swings live or by studying them using the video analysis tools he normally uses.

In summary, the contributions of this work are:

1. Presenting a new type of visualization problem called video diffing, in which subtle motion differences between videos are highlighted.
2. Developing a method to measure video dissimilarity using spatiotemporal gradients evaluated at different scales.
3. Presenting a way to display the differences of two videos at once by overlaying edges of one video onto another.
4. Empirical evaluation with a user study and anecdotal results.

2 Related Work

Our work is related to research fields such as motion magnification, video alignment, action recognition and pyramid matching. It is also related to commercial applications that coaches currently use for comparing video recordings in sports.

Our work is inspired by recent papers on motion magnification [Wadhwa et al. 2013; Wu et al. 2012], in that we wish to highlight motions in video. But while those works focus on magnifying imperceptible motions in one video, we highlight differences between multiple videos. We do not magnify motions but make differences more apparent. We considered magnifying differences, but concluded that this was more likely to be misleading than helpful, since the absolute degree of difference can be important. The motion magnification work assumes all motion is small while we are interested in small differences between big motions.

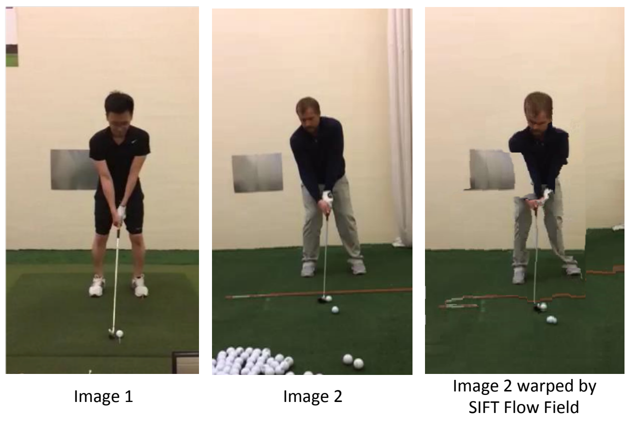

Figure 3: *An example of using SIFT Flow on a pair of images of two golfers. The third image is the result of warping Image 2 by the SIFT Flow field. A good field would make the warped image look similar to Image 1 in positioning and orientation. However, there are large errors with the arms and legs.*

There is a rich field of research related to video alignment [Padua et al. 2008; Evangelidis and Bauckhage 2013; Caspi and Irani 2002; Ukrainitz and Irani 2006; Sand and Teller 2004; Diego et al. 2013]. Alignment focuses on spatiotemporally registering videos of the same scene captured from different viewpoints and times with static and moving cameras. In contrast, we do not find a dense alignment between videos. While such an alignment could indeed help in visualizing differences, the methods are often restricted to videos of the same scene. We compute visualizations of two different scenes that may contain different people/objects. We found that dense matching is not robust to different scenes. Even an algorithm like SIFT Flow [Liu et al. 2008], which is intended for this purpose often yields inaccurate correspondences. Fig. 3 shows frames of two different golfers in a similar position along with an image displaying how well the SIFT Flow field warps the second image to the first. There is a large error for the arms and legs. Finally, our purpose to produce an effective visualization as opposed to just finding the alignment is also fundamentally different from all these previous works.

Our approach compares input videos using histograms of spatiotemporal gradients. Analogous to 2D HOG [Dalal and Triggs 2005], which bins spatial gradients, there are 3D histogram features proposed in the computer vision literature [Kläser et al. 2008]. Spatiotemporal gradient features have previously been used in action recognition research [Laptev 2005; Dollar et al. 2005]. We use histograms of gradients to obtain a measure of dissimilarity at each pixel between the videos. The dissimilarity is then used to better visualize differences instead of performing a classification task.

Our work builds on the idea of spatial pyramid matching and pyramid match kernels [Lazebnik et al. 2006; Grauman and Darrell 2007]. In spatial pyramid matching, an image is partitioned into increasingly fine subregions from which histograms are computed. Histogram distances are weighted by the scale of their regions and summed to obtain a distance between images. This type of approach is inspired from the pyramid match kernel for feature spaces. Unlike this work, we do not combine histogram distances from different spatial regions into one score. We compute histograms at different spatial scales to assign a score to each pixel between the videos.

We borrow the idea of using spatial edges in a visualization from computation re-photography research [Bae et al. 2010]. In that work the authors introduce a visualization technique using edges that helps users reach a desired viewpoint during capture. Specifically, edges assist the user to evaluate the match between a reference image and a new scene.

Several commercial apps focus on improving athletic performance with video. Some of the prominent names are Ubersense, Coach's Eye and Dartfish Express [Ubersense 2015; Coach's Eye 2015; Dartfish 2015]. They all contain a side-by-side comparison feature that allows, for example, golfers to compare different versions of their own swings, or their swings with the swings of others.

3 Method

We take two grayscale input videos $A(x, y, t), B(x, y, t) : \mathbb{R}^3 \rightarrow \mathbb{R}$ and output a RGB video $V(x, y, t, c) : \mathbb{R}^4 \rightarrow \mathbb{R}$ that highlights their differences. (x, y, t) is a location in spacetime and $c \in \{0, 1, 2\}$ represents a RGB channel number. A and B are recordings of similar objects/people performing a similar action. We assume for ease of explanation that the videos share the same height, width, and number of frames. We assume for now that the videos are prealigned in space and time. If this is not the case, the videos can be quickly prealigned as described in Section 3.3.

Our output V overlays the spatial gradients (edges) of B over the frames of A. The edges of B are colored based on a map $W(x, y, t) : \mathbb{R}^3 \rightarrow \mathbb{R}$ that stores the spacetime disimilarity of the videos at each pixel location.

Fig. 4 gives an overview of our method. We extract the spatiotemporal gradient of each pixel of A and B. For each frame, we form a pyramid of gradient descriptors, with each pyramid level consisting of descriptors coarser in spatial resolution than the previous level. Using the distances between the descriptors of the pyramids, we compute a dissimilarity map W. Finally, we use W, A and edges of B to form the output V. Edges are marked in color based on W to draw the user's attention to significant differences between the inputs.

3.1 Forming the Dissimilarity Map

A main challenge in this work is to find potentially relevant dissimilarities between videos even when the foregrounds are not the same. We compute a dissimilarity map W between A and B to do this by extracting spatiotemporal gradients and computing distances between descriptors of these gradients at different spatial scales. We combine the distances at different scales into one score per pixel that indicates how dissimilar the two videos are at that pixel.

3.1.1 Spacetime Gradient Extraction and Quantization

The low-level information we use from A and B are spatiotemporal gradients at each pixel. Common alternatives to spatiotemporal gradients described in the action recognition literature are pixel intensities or a higher level of abstraction such as optical flow. A comparison of these different sources of information indicate that gradients and optical flow are most informative [Wang et al. 2009]. We did not use optical flow because dense flow algorithms can be brittle and only measure motion and disregard local appearance characteristics. If for example, two body parts are misaligned between the videos but are moving with the same velocity, optical flow will not report a difference. The spatial components of 3D gradients can differentiate between dissimilar structures.

For each pixel $\mathbf{x}$ of A (and similarly for B) we extract a gradient vector $\bar{\mathbf{g}}_{\mathbf{x}}$ consisting of the first-order partial intensity gradients: $(\frac{\partial A(\mathbf{x})}{\partial x}, \frac{\partial A(\mathbf{x})}{\partial y}, \frac{\partial A(\mathbf{x})}{\partial t})$. To compute the gradients, we take central differences of neighboring pixels. Similar to past work [Kläser et al. 2008], we quantize a gradient $\bar{\mathbf{g}}$ by orientation into a histogram using a 3D icosahedron. Each bin of the resulting histogram corresponds to one of the icosahedron's faces. Let P be the matrix of center positions $\mathbf{p_1}, \cdots, \mathbf{p_n}$ of all 20 faces, where the center positions are defined by:

$$(\pm 1, \pm 1, \pm 1), (0, \pm\frac{1}{\phi}, \pm\phi), (\pm\frac{1}{\phi}, \pm\phi, 0), (\pm\phi, 0, \pm\frac{1}{\phi}) \quad (1)$$

where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio. The projection $\hat{\mathbf{q}}$ of $\bar{\mathbf{g}}$ is obtained through:

$$\hat{\mathbf{q}} = (\hat{q}_1, \cdots, \hat{q}_n)^T = \frac{P \cdot \bar{\mathbf{g}}}{||\bar{\mathbf{g}}||_2} \quad (2)$$

The resulting 20-bin histogram is directed, meaning that gradients with opposite orientations are placed into separate bins. We form an undirected histogram of 10 bins by halving the set of face centers and taking the absolute value of $\hat{q}_i$.

The vector $\hat{\mathbf{q}}$ is thresholded so that $\bar{\mathbf{g}}$ only votes in a single bin in case it is perfectly aligned with one of the face centers. We subtract a threshold $t = \mathbf{p_i}^T \cdot \mathbf{p_j} = 1.29107$ from $\hat{\mathbf{q}}$ and set all negative

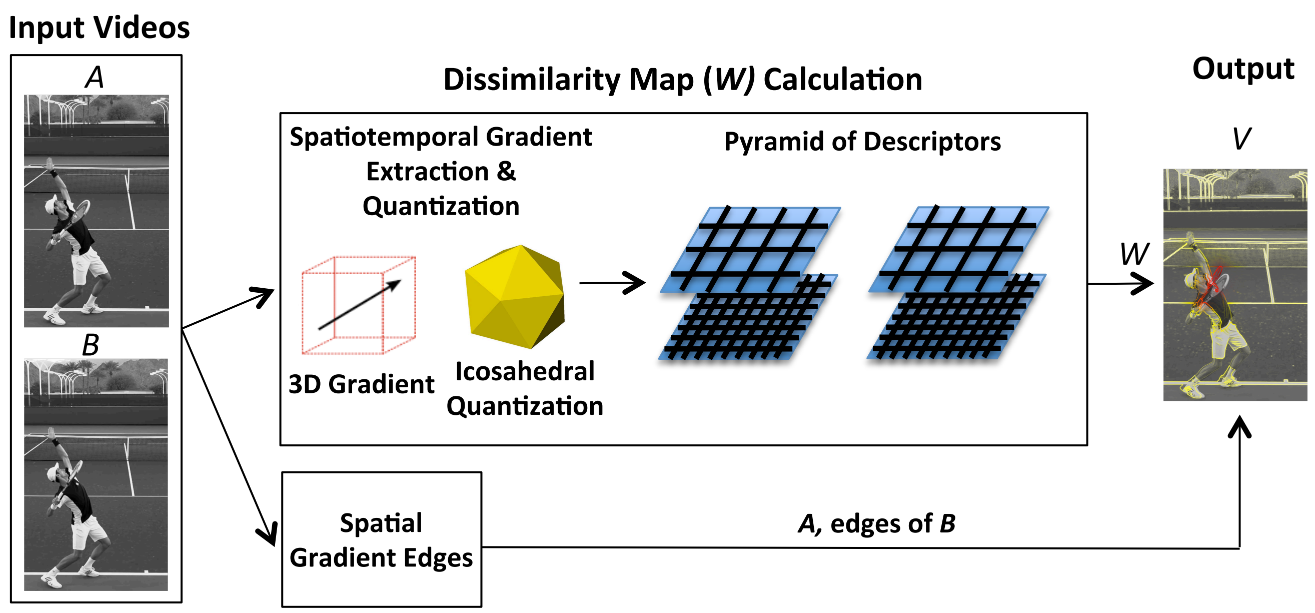

Figure 4: *Overview of our method. Two gray-scale videos A and B are given as input. We extract spatiotemporal gradients from both videos, and calculate histograms of these gradients at different spatial scales on a frame-by-frame basis. We form a dissimilarity map W by computing a weighted (by level) sum of distances between the pyramids. Finally, we overlay edges of B on frames of A, using W to colorcode the edges appropriately. Tennis images courtesy of Essential Tennis on YouTube.*

elements to zero. Finally, the gradient magnitude is distributed according to the thresholded histogram $\hat{\mathbf{q}}'$:

$$\mathbf{q} = \frac{||\bar{\mathbf{g}}||_2 \cdot \hat{\mathbf{q}}'}{||\hat{\mathbf{q}}'||_2} \tag{3}$$

We do not quantize every pixel of the input videos. As described in the next section, we only quantize the average gradient in blocks of the image based on the location of the descriptors we extract.

3.1.2 Descriptor Computation

We measure local dissimilarity by calculating the difference between descriptors of the spacetime gradients. Our descriptors are extracted over varying spatial scales and a fixed temporal scale of 1 frame. Adding variation in temporal scale requires us to determine the relative importance between spatial and temporal scale differences in the visualization which varies for different applications.

A descriptor $\mathbf{d}$ in frame t with local support (x, y, s) is computed over a square of side s pixels centered at pixel (x, y). We divide the square into a grid of M x M cells $\mathbf{c_1} \cdots \mathbf{c_{M^2}}$. We compute a histogram $\mathbf{h_{c_i}}$ for each cell and concatenate all histograms to form the final descriptor $\mathbf{d} = (\mathbf{h_{c_1}}^T, \cdots, \mathbf{h_{c_{M^2}}}^T)^T$. To calculate $\mathbf{h_{c_i}}$, we further divide the cell $\mathbf{c_i}$ into a m x m grid of blocks. We compute the average spatiotemporal gradient of each block, quantize the gradient into a 10-dimensional histogram as described in 3.1.1, and compute a weighted sum of the histograms of the m^2 blocks. We weight each block's histogram by the distance of the block to (x, y) using a 2D gaussian kernel.

The descriptor $\mathbf{d}$ is of dimension $10M^2$. We normalize $\mathbf{d}$ with a regularized L_2 norm: $||\mathbf{d}||_2 + 1$ to make the descriptor invariant to illumination changes. We set $M = 2, m = 4$ for all our videos.

3.1.3 Pyramid of Descriptors

We build on pyramid matching [Lazebnik et al. 2006; Grauman and Darrell 2007] and use a multiscale approach where differences in descriptors at coarser spatial scales are weighted more heavily than those at finer scales.

We compute the values of W for each frame independently. For frame t, we place a sequence of grids of increasingly coarse descriptors. We call this sequence of grids a pyramid. Let L be the number of levels in the pyramid (ranging from 0 to $L - 1$), which we set at 3 for our videos. At level l, we use descriptors of scale $s_l = 2^l s_o$, where s_0 is set by the user. We set s_0 to 32 for our videos. The descriptor centers are located in a grid every $\frac{s_l}{Mm}$ pixels, meaning that descriptors overlap with their neighbors.

We calculate the distances between corresponding descriptors of the two videos for each level:

$$W^{t,l}(x, y) = ||D_A^{t,l}(x, y) - D_B^{t,l}(x, y)||_2 \tag{4}$$

where $W^{t,l}$ contains the dissimilarities computed for level l of frame t and $D_A^{t,l}(x, y), D_B^{t,l}(x, y)$ are the descriptors at location (x, y, t) in level l of the pyramid. $W^{t,l}$ is sparse since dissimilarities are only computed at descriptor centers. We bilinearly interpolate the missing values. We next combine the dissimilarities from each level into one dissimilarity value per pixel. We use a weighted sum over all levels to compute W^t, the map for frame t:

$$W^t = \sum_{l=0}^{L-1} 2^l W^{t,l}(x, y) \tag{5}$$

Coarser levels are weighted more heavily because they correspond to larger-scale differences. We set the weight of each level proportional to its scale, similar to what is done in pyramid matching

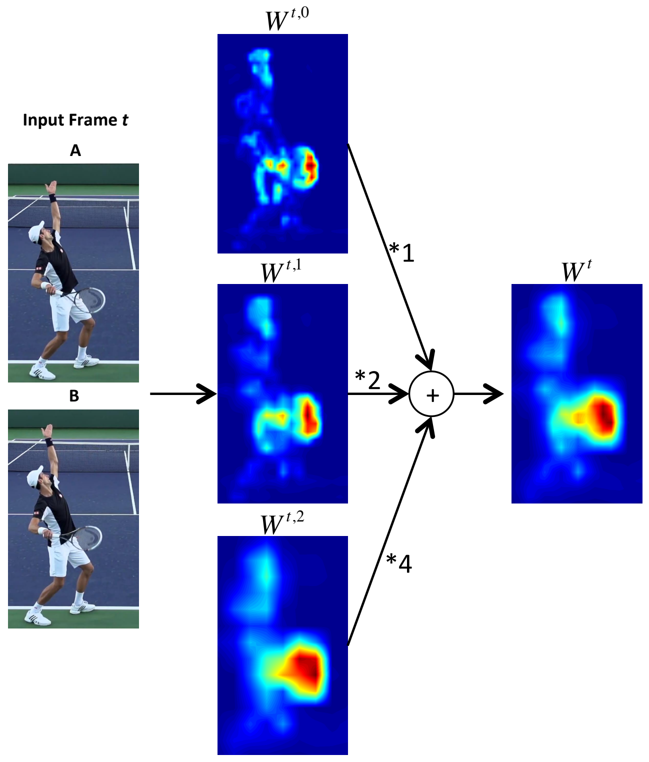

Figure 5: *Example of computing W^t, the weight map for frame t. Weight maps computing differences at various scales are first computed and then added together in a weighted sum to produce W^t.*

kernels and spatial pyramid matching. Fig. 5 shows an example of producing W^t for a particular frame t of our tennis example.

If A, B are shot in different environments or even if the backgrounds are spatially offset from one another, W will contain constant, nonzero values at background pixels. This is problematic because our goal is to emphasize differences in the action sequence and not differences in the environment. To address this we temporally highpass W to deemphasize constant or slowly-changing values.

Finally,we normalize W by its maximum value over all frames so that all values lie in the range $[0, 1]$. A value of 1 indicates the most dissimilar pixel in the video.

3.2 Producing the Overlay

Let K be a RGB heat map of W, formed with some heat map function $f_c : \mathbb{R} \in [0, 1] \to \mathbb{R}^3$ that takes a dissimilarity score and returns a RGB value. Fig. 6 shows the f_c that we used for all our videos.

We form V by blending K and A, weighted by the spatial gradient of B. First, we compute the spatial gradient magnitude $G(x, y, t)$:

$$G(x, y, t) = \min\left(1, \alpha\sqrt{B_{\partial x}(x, y, t)^2 + B_{\partial y}(x, y, t)^2}\right) \tag{6}$$

where α is a factor that can reduce or emphasize gradients depending on how strong they are in the input video pair. We found that using the Sobel operator here, along with $\alpha \in [0.5, 1]$ worked best.

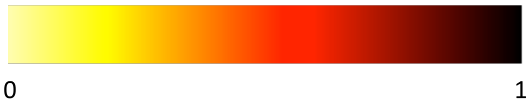

Figure 6: *The heatmap function f_c we used for our videos. Yellow corresponds to colors of low dissimilarity and dark red for larger ones.*

Values are clipped at 1 since this is the maximum value of any output pixel.

We then blend K and A:

$$\begin{aligned} V(x, y, t, c) = &(1 - G(x, y, t)) \cdot A(x, y, t) + \\ &G(x, y, t) \cdot K(x, y, t, c) \end{aligned} \tag{7}$$

where c is the channel number. Depending upon the function f_c that is used, it may be necessary to reduce the contrast of A to improve the visualization. For example, given the colormap shown in Fig. 6, we found it best to remove the very light and very dark intensities of A. Our algorithm reduces the contrast of A on a frame-by-frame basis. For each frame we map the value at the first percentile of pixel intensities I_{low} to β_{low} and the value at the 99th percentile I_{high} to β_{high}, where $\beta_{low}, \beta_{high} \in [0, 1]$. All pixel intensities are appropriately scaled, and values outside the range $[\beta_{low}, \beta_{high}]$ are clipped.

We set $\beta_{low} = 0.2$ and $\beta_{high} = 0.8$ for all our videos. Tunable variables in the overlay process include $\alpha, \beta_{min}, \beta_{max}$ and the color function f_c. We kept the same settings for all our videos, but a user can test different parameters quickly given a precomputed W. Fig. 7 shows the effect of adjusting these parameters for one frame of our tennis example.

The overlay process is not symmetric. That is, there is a difference in output depending on which video is chosen as the reference (A) and which is not. Which looks better depends on the specific example, but in general a video with prominent foreground edges and few background edges is an ideal choice for video B.

3.3 Prealignment

Our visualization is most useful when the input videos are already aligned in space and time. If the two videos are not aligned, we bring them into alignment by automatically fitting a translational temporal transformation and then manually fitting an affine spatial transformation. We do not perform a dense spatiotemporal alignment, since this would remove the differences that the user wants to observe in the first place. Unlike past works [Caspi and Irani 2002; Ukrainitz and Irani 2006], we do not jointly optimize the alignment along the spatial and temporal dimensions. Instead, we first temporally align the sequences without any spatial alignment, and then perform the spatial alignment.

We compute temporal alignment using the same spatiotemporal gradient quantization we used for the visualization (see Sec. 3.1.1). We represent each frame t of video A (and similarly for B) with a 10-dimensional vector $\mathbf{q_A^t}$ that we calculate by adding together the quantizations of all pixels in the frame. We encode no information about the location of each gradient, since the videos are not yet spatially aligned. The sequence of vectors $\mathbf{q_A^0}, \mathbf{q_A^1}, \cdots \mathbf{q_A^{T-1}}$ form a

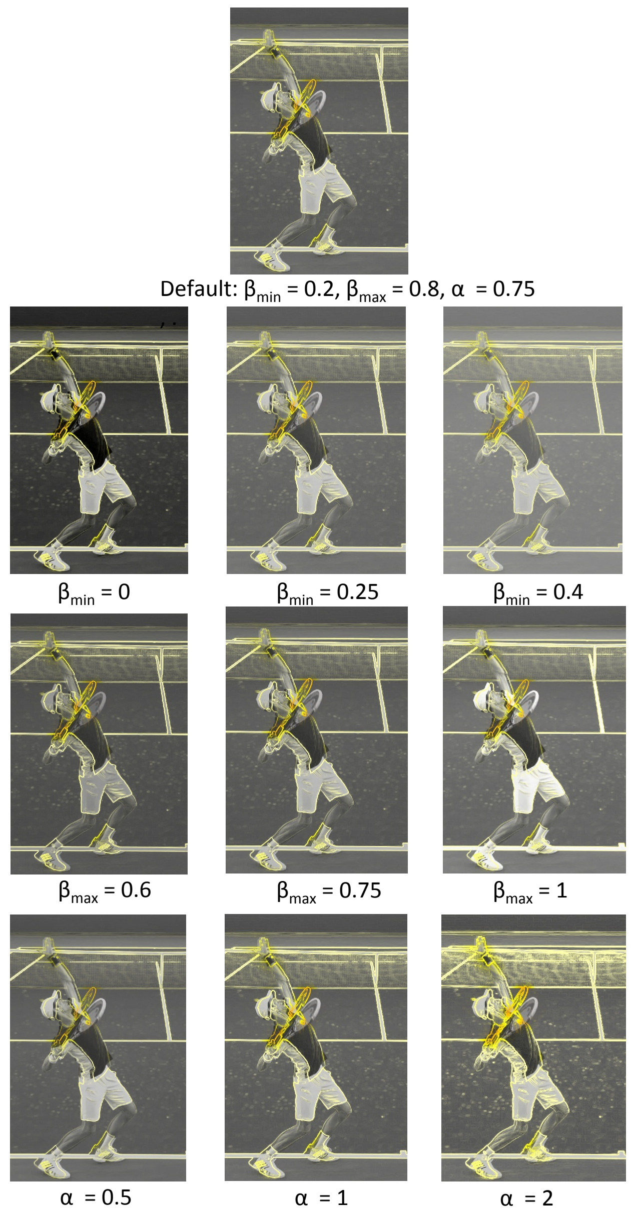

Figure 7: *Overlays showing the effect of adjusting the parameters α, β_{min} and β_{max}. The default setting is $\alpha = 0.75, \beta_{min} = 0.2, \beta_{max} = 0.8$. Each parameter is adjusted while holding the others constant. Row 2 adjusts β_{min}, row 3 adjusts β_{max} and row 4 adjusts α.*

time series for video A. To align the videos, we bring their time-series into translational alignment, assuming the videos are shot at a similar frame rate and that the actions are of similar duration. We do this by simply calculating a normalized cross-correlation for all possible integer offsets of the time series and choosing the maximum. To also accomodate scale transformations, one can use Newton's method to find optimal parameters, similar to previous work for full spatiotemporal alignment [Ukrainitz and Irani 2006].

We next perform spatial alignment. For our videos we found that it is more useful to spatially align the beginning of the sequences rather than the later parts. For example, spatially aligning two tennis serves based on the starting position leads to a more intuitive

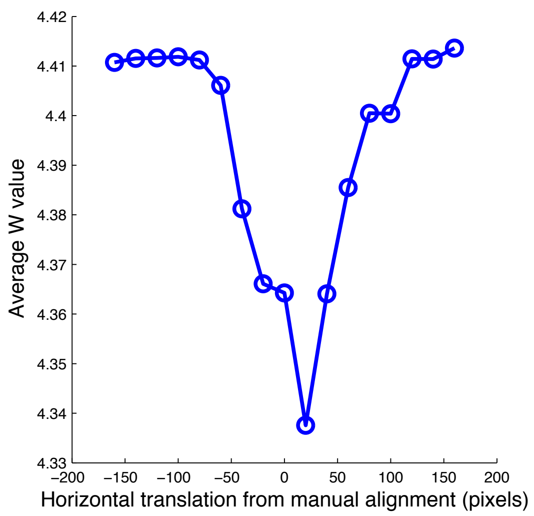

Figure 8: *The average dissimilarity value of W over all pixels when translating video B by different horizontal shifts from the manual prealignment. Minimum dissimilarity occurs at a translation close, but not equal to 0. This is likely because the later parts of the swings are offset from one another.*

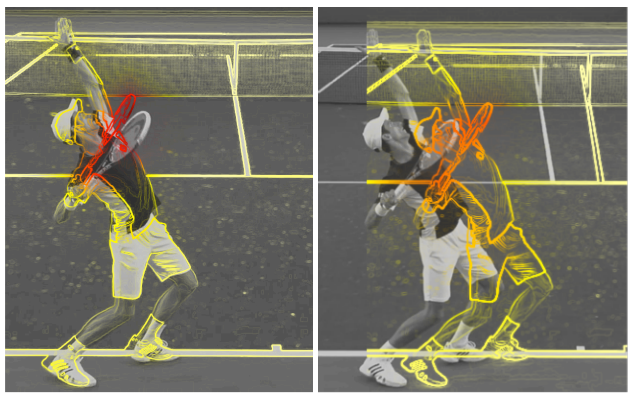

Figure 9: *An example of an overlay when the foregrounds are aligned (left) versus not aligned (right). In the aligned case, the racket is clearly seen to be the major difference. In the misaligned case, parts of the body are also highlighted.*

visualization than aligning by the moment of ball impact when the bodies may diverge. We align the sequences based on their first frames (as determined by the temporal alignment). As Fig. 3 shows, matching image features across frames is not reliable with different foreground objects, even when the scenes have very similar backgrounds. Instead, we let the user select a few foreground point correspondences and fit a 2D affine transformation matrix to the point pairs. All frames of the second video are spatially transformed by this matrix.

Our algorithm will produce a visualization overlay regardless of how well the prealignment phase works. The strength of the spatial edges of B are unaffected, although they become less useful with larger misalignment. The color of the edges as determined by the weight map W will be altered. Fig. 8 shows how the average value of W for our tennis example changes when B is translated horizontally by different amounts. Minimum dissimilarity occurs at a horizontal translation close, but not equal to, 0 because the later parts of the swings deviate from one another. Fig. 9 shows an overlay when B is translated 100 pixels to the right. In the aligned case, the racket is clearly seen to be the major difference. In the misaligned case, parts of the body are also highlighted leading to a misleading overlay.

4 Experiments and Results

We performed a user study designed to compare the ease with which a viewer could find differences between a pair of videos using our approach compared to each of the two baselines described below.

We selected three different video pairs from Youtube:

1. Two practice serves of Novak Djokovic. The serves are aimed at different locations in the service box and possibly have different spins.
2. Two successful clean-and-jerk attempts from the same lifter in a competition. The lifts are with different weights.
3. A pair of drives, one from Tiger Woods and the other from Rory McIlroy.

Videos are courtesy of the Essential Tennis, Karumor and GolfswingHD channels on Youtube. Fig. 10 shows example output frames for the weightlifting and golf videos. Example output frames for the tennis videos are shown in Fig. 7.

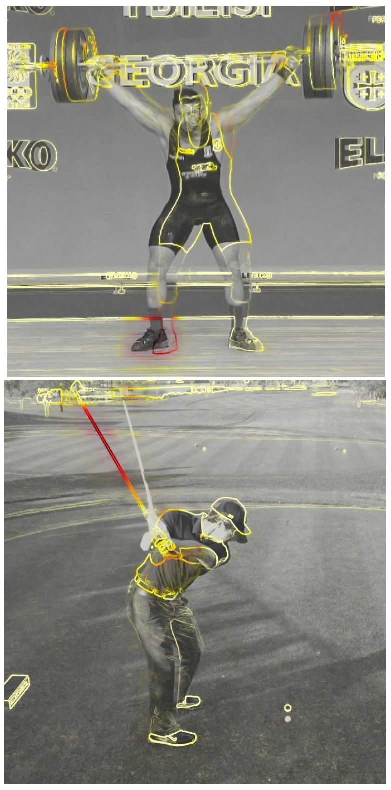

Figure 10: *Output frames from our weightlifting and golf survey videos.*

We evaluated three methods for comparing these videos: side-by-side (SS), false color channel overlay (CO), and our video diff algorithm (VD). CO is constructed by placing the grayscale frames of one video into the R and B channels of the output and the grayscale frames of the second video in the G channel. Areas that appear magenta and green are different, while areas in grayscale are the same. For VD and CO, we also show the input video pair next to the method's outputs for context. An example output of CO is shown in Fig. 2b.

Subjects were asked to find as many differences as they could between each pair within 3 minutes. We did not suggest spatial or temporal areas to focus on, but did tell them to limit their observations to the foreground of each scene. Each user was shown the output of only one method for each video pair. Each subject saw each method and each pair once, with the order of the methods and pairs randomized. After completion of the tasks, they ranked each method on ease of interpretability on a scale of 1 (very hard to interpret) to 5 (very easy) and were also asked for any other comments/feedback.

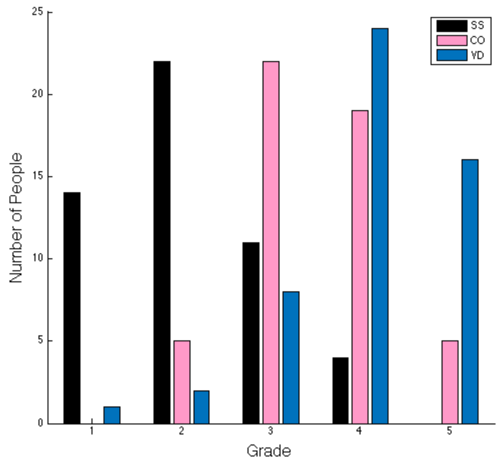

Figure 11: *Results from the user study. Participants graded the three methods (SS (side by side), CO (channel overlay) and VD (our video diff algorithm)) on a scale of 1-5, with 5 indicating that the method makes visualizing differences very easy and 1 indicating that the method is very hard to interpret. Subjects graded VD the best with an average score of 4.04. CO followed with 3.48 and SS was last with 2.08.*

We recruited 54 participants for the study. The participants were men and women $20-35$ years old. Their expertise in sports ranged widely, but none were professional coaches or athletes. The participants took the user study online, without administration or help from the authors. Fig. 11 shows the distribution of user grades for the three methods. Users graded VD with an average score of 4.04 out of 5.0, as opposed to 3.48 for CO and 2.08 for SS. Using the Kolmogorov-Smirnov (KS) test, we can reject the null hypotheses that SS and CO are from the same distribution ($p < 10^{-9}$) and that CO and VD are from the same distribution ($p < 0.01$).

Table 1 shows the average and standard deviation of grades broken down by video pair and method. Numbers in parentheses indicate the size of the sample. The average grade for VD was higher than CO and SS for all three video pairs. VD's improvement over CO was smallest for pair 3. We speculate this was because the golfers in this pair were different, making it harder for users to compare their swings.

When examining the type of differences reported, we found that timing differences (e.g., swing speed in tennis and golf) and subtle spatial differences (e.g., the swing path of the golfers) were discussed more when using CO and VD than SS. We found no obvious

Pair	Method SS	CO	VD
1	$2.06 \pm 0.87(18)$	$3.59 \pm 0.94(17)$	$4.16 \pm 0.83(19)$
2	$2.27 \pm 1.10(15)$	$3.25 \pm 0.72(20)$	$4.16 \pm 0.69(19)$
3	$1.90 \pm 0.77(21)$	$3.59 \pm 0.71(17)$	$3.75 \pm 1.13(16)$

Table 1: *Average grade ± the standard deviation for each method/video pair combination. Numbers in parentheses are the number of samples. VD performed better than SS and CO for all three video pairs.*

qualitative differences in the comments of the CO and VD. We also found no significant difference between the number of differences found using each method. The quality and quantity of comments were more linked to the expertise of viewers (determined based on vocabulary used and feedback to the survey questions) than on anything else. Additionally, many users felt that the time limit of 3 minutes limited the number of differences they could report.

4.1 Feedback from a Professional Golf Coach

We met with a PGA (Professional Golf Association)-certified golf coach to determine whether our overlays would be useful in an instructional setting. The coach regularly takes video recordings of his clients during lessons, and uses the video both to instruct and to analyze techniques. We obtained several of his video recordings of a novice golfer that the pro shot using an iPhone camera.

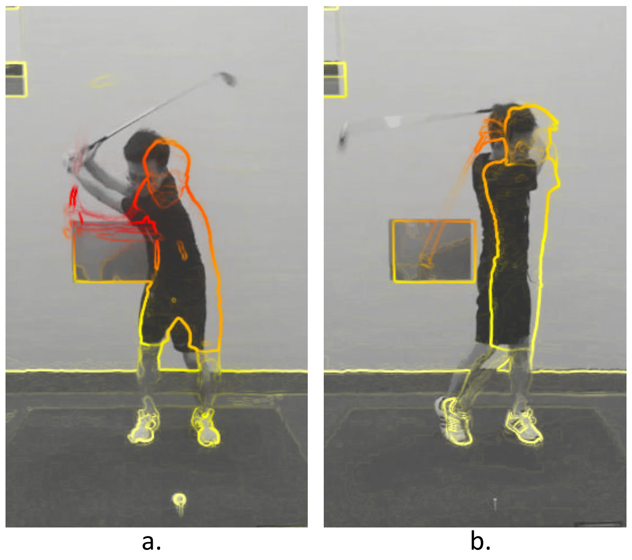

Figure 12: *Sample frames of overlays for videos of a golf student. The golf instructor noted how the student had a large variance in motions across his swings. In (a.), his left knee moves differently, causing movement in his entire left side. In (b.), the follow-through of the arms are inconsistent.*

The coach considered our overlays "eye-opening" in that he did not realize how much swing-to-swing variation there was. Fig. 12 shows overlay frames for two pairs of the student's videos. After examining the overlay video of the first pair, the pro observed that there was a notable difference in both the "the speed and timing of the backswing" (not shown in the figure) and that "the timing of the lower body moving on the way through the ball, especially the left knee, indicated a lot about his lack of ability to use the lower body properly." (Fig. 12(a)). For the second pair of swings, the instructor found that the differences in arm movement after hitting the ball during follow-through were significant (Fig. 12(b)). He stated that all of these differences can lead to major inconsistencies in results.

4.2 Other Videos

Fig. 13 shows sample frames from several other videos on which we tested our method. Figs. 13(a,b) shows overlays for synthesized videos of animal locomotion used in a recent publication [Wampler et al. 2014]. The goal of that work was to predict the gait of animals based on their shapes. The authors compared videos of the outputs of their algorithm to videos based on tracking data (ground truth). We ran our method on their pairs and showed them the output. They pointed out several differences that they felt would have been hard to notice with the side-by-side video alone. For the video depicted in Figure 13(a) of an elephant simulation, the authors commented that the spacing between the front and back legs is slightly larger for ground truth. Observing the overlay of an ostrich simulation, depicted in Fig. 13(b), they noticed that the legs of their model lag behind the legs of ground truth during the swing phase.

Fig. 13(c) shows an overlay of two ballet dancers trying to dance in synchrony. In this frame, we see that the dancer of the edge video has his right arm a little lower than the other dancer. Fig. 13(d) is an overlay of an individual kicking a soccer ball twice. In the edge video, the kicking foot rotates a little further during contact. Fig. 13(e) depicts two fastballs by Major League Baseball pitcher, Alex Wilson. The back leg of the edge video is lifted higher after releasing the ball. Finally, Fig. 13(f) is of two deformed rods vibrating back and forth. There is a slight difference in the amplitude of the rod vibrations as shown in this frame.

4.3 Processing Time

Table 2 gives the runtime of our method (without prealignment) for several videos of varying size. We ran our algorithm on a Macbook laptop with a 2.6 GHz processor and 16 GB of memory. The code was written in a mixture of Matlab and mexed C code. The processing times show that our method is almost real-time for small videos and within a few minutes of real-time for relatively large videos. Although not done here, processing time can be reduced via parallelization, since our dissimilarity map W is computed independently for each pair of input frames.

Video	Frame Size	Number of Frames	Time(sec)
Student's Golf Swings	550x300	240	27
Basketball	1024x496	296	92
Djokovic Tennis Serves	1080x650	403	218
Woods vs. McIlroy Golf Swings	864x728	545	302

Table 2: *Runtimes for videos of different sizes.*

5 Limitations

A limitation of our method is that dissimilarity is based only on local intensity-gradient distributions. This can be misleading because even if two patches have similar gradient distributions, they are not necessarily the same region of the object/body of interest. For example, in Fig. 13(c), the dancer's whole left arm in the second video is translated to the right (in the image plane), but our visualization only indicates the location change of the hand. In the future we would like to incorporate matching into our visualization to present richer information to the user. But as shown earlier, matching algorithms like SIFT Flow can lead to misleading results, which must be handled appropriately to not ruin the visualization.

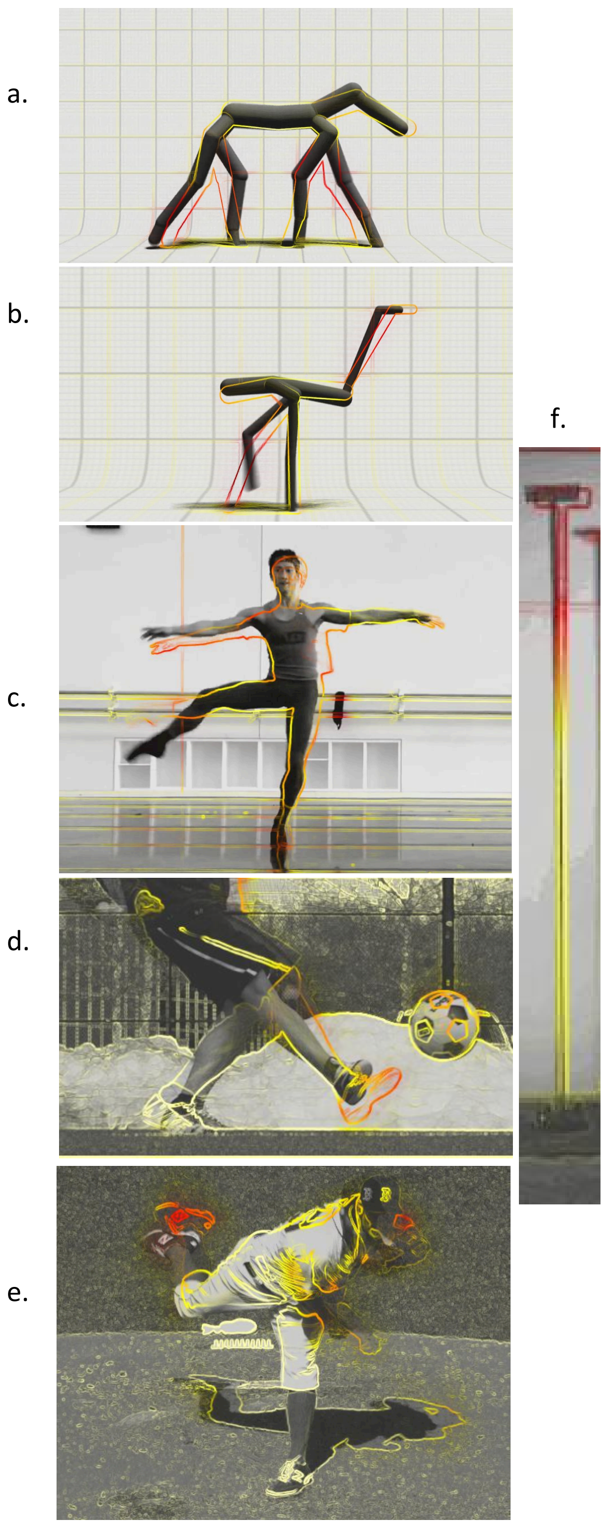

Figure 13: *Outputs for various examples. Refer to the text for an explanation. (a.) and (b.) are of animal locomotion videos in a recent work [Wampler et al. 2014]. (c.) is of two ballet dancers. (d.) is of the same person kicking a soccer ball twice. (e.) depicts a Major League Baseball player throwing two pitches. (f.) is of two deformed rods vibrating back and forth.*

Several subjects in the user study mentioned that better time synchronization would have helped them focus on non-timing related differences. For example, in the two videos of Novak Djokovic's serves, the moment of impact with the ball was not the same. Some users said that Djokovic's body motions may have been similar at impact but it was not easy to see this since they occurred at different frames. The misalignment is a result of our prealignment optimization where we constrained the temporal alignment to a 1D affine transformation. We could have allowed for a more complex alignment in time. But while it may be useful to perform video diffing by removing temporal differences first, in many cases the temporal differences are important to the task. For example, the fact that the racket speed was greater in one video than the other is highly relevant. And as the golf instructor stated, consistency of the speed of the swing is important. Because of this, it is likely that two independent video diff techniques, one that compensates for timing differences and one that does not, will be needed.

Another limitation is that we treat the entire frame as the foreground. Because of this, edges in the scenery of the second video will be present in the overlay and can cause confusion when superimposed on the foreground of the first video. Examples of this include the background trees and fence in Fig. 13(d), and the shadow and small gradients in the sand/grass in Fig. 13(e). A useful future direction would be to estimate the likelihood that each pixel belongs to the foreground or background, and adjust the overlay by removing edges that are certainly part of the background. In a similar vein, our method can produce bad visualizations when the camera of either video is moving substantially. This is because there will be dissimilar temporal gradients between all pixels of the videos. Stabilization and/or foreground/background subtraction may be needed to account for this.

Finally, what is highlighted in our visualizations are the motions projected on the image plane. There is no quantification of the variation of depth in the foreground's position or motion. This is important to remember, since for example, depth variations of the hands can lead to missed shots in basketball and golf even though they may look extremely similar in the image plane.

6 Summary

We presented video diffing, a new type of problem in which we highlight subtle action differences between a pair of videos. Our video diff algorithm first finds the dissimilarity between motions using a novel video dissimilarity metric based on spatiotemporal gradients evaluated at different scales. We then provide a visualization of the difference by overlaying the edges of one video over the frames of the second video. We color the edges based on the level of local dissimilarity between the input videos. Our method is simpler than a full spatiotemporal optimization, which makes it robust to many different types of scenes and quick to process.

We evaluated the method with a user study with 54 subjects. The results from the user study indicate that most users preferred our overlay to side-by-side videos or merging the videos into different color channels. A professional golf instructor also stated that he thinks our visualizations can be used in numerous ways throughout a lesson and in the study of students' progression over time.

Acknowledgments

This work was funded by Qatar Computing Research Institute (QCRI) and Quanta Computer.

References

BAE, S., ET AL. 2010. Computational rephotography. *ACM Trans. Graph. 29*, 3, 24:1–24:15.

CASPI, Y., AND IRANI, M. 2002. Spatio-temporal alignment of sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 11, 1409–1424.

COACH'S EYE, 2015. `https://www.coachseye.com`.

DALAL, N., AND TRIGGS, B. 2005. Histograms of oriented gradients for human detection. 886–893.

DARTFISH, 2015. `http://www.dartfish.com/Express`.

DIEGO, F., ET AL. 2013. Joint spatio-temporal alignment of sequences. *IEEE Transactions on Multimedia 15*, 6, 1377–1387.

DOLLAR, P., RABAUD, V., COTTRELL, G., AND BELONGIE, S. 2005. Behavior recognition via sparse spatio-temporal features. 65–72.

EVANGELIDIS, G. D., AND BAUCKHAGE, C. 2013. Efficient subframe video alignment using short descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 10, 2371–2386.

GRAUMAN, K., AND DARRELL, T. 2007. The pyramid match kernel: Efficient learning with sets of features. *J. Mach. Learn. Res. 8*, 725–760.

KLÄSER, A., MARSZAŁEK, M., AND SCHMID, C. 2008. A spatio-temporal descriptor based on 3d-gradients. 995–1004.

LAPTEV, I. 2005. On space-time interest points. *Int. J. Comput. Vision 64*, 2-3, 107–123.

LAZEBNIK, S., ET AL. 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2169–2178.

LIU, C., ET AL. 2008. Sift flow: Dense correspondence across different scenes. *Proceedings of the 10th European Conference on Computer Vision: Part III*, 28–42.

PADUA, F. L., ET AL. 2008. Linear sequence-to-sequence alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 2, 304–320.

SAND, P., AND TELLER, S. 2004. Video matching. *ACM Transactions on Graphics 22*, 3, 592–599.

UBERSENSE, 2015. `http://www.ubersense.com/`.

UKRAINITZ, Y., AND IRANI, M. 2006. Aligning sequences and actions by maximizing space-time correlations. *ECCV (3)*, 538–550.

WADHWA, N., ET AL. 2013. Phase-based video motion processing. *ACM Trans. Graph. (Proceedings SIGGRAPH 2013) 32*, 4.

WAMPLER, K., ET AL. 2014. Generalizing locomotion style to new animals with inverse optimal regression. *ACM Trans. Graph. 33*, 4, 1–11.

WANG, H., ET AL. 2009. Evaluation of local spatio-temporal features for action recognition. 124.1–124.11.

WU, H.-Y., ET AL. 2012. Eulerian video magnification for revealing subtle changes in the world. *ACM Trans. Graph. (Proceedings SIGGRAPH 2012) 31*, 4.