CONTEXTUAL POINT MATCHING FOR VIDEO STABILIZATION

LING HAN MENG, JOSEPH GEUMLEK, HOLLY CHU, JUSTIN HOOGENSTYRD
ADVISOR: ERNIE ESSER
UNIVERSITY OF CALIFORNIA, IRVINE
NSF PRISM GRANT DMS-0928427

Abstract.

We explore the potential of applying a contextual shape matching algorithm to the domain of video stabilization. This method is a natural fit for finding the point correspondences between subsequent frames in a video. By using global contextual information, this method outperforms methods which only consider local features in cases where the shapes involved have high degrees of self-similarity, or change in appearance significantly between frames while maintaining a similar overall shape. Furthermore, this method can also be modified to account for rotationally invariant data and low frame rate videos. Though computationally-intensive, we found it to provide better results than existing methods without significantly increasing computational costs.

1. Introduction. Many videos suffer from a shaking or moving camera, and these camera movements tend to only distract the viewer from the subject matter. However, these movements can be fixed in software, in a process known as video stabilization. Many methods focus on point correspondences, where points from one frame of the video are matched with points in the next frame[2, 9, 13, 3]. If a transformation that aligns the points from one frame with their corresponding points in the other frame can be found, then the underlying video frames can be aligned using the same transformation. Points are matched using descriptors, which quanitify characteristics of the points, so that points with the best matching descriptors are the most likely to be matching on the underlying frames. These descriptors are traditionally based on local features, which are the properties of the region of the frame surrounding a point. We have borrowed the descriptor outlined by Belongie, Malik and Puzicha[4] which describes a point using the distribution of points around it. This descriptor differs fundamentally from the local feature descriptors in that is contextual; it depends not on the actual traits of the video frame, but rather on the set of key points selected from the frame. It is also more global than local, since it considers points far away from a given point in computing the descriptor rather than just a small image patch under the point. This global contextual descriptor was originally used to find point correspondences for recognizing shapes against a library of templates, but our work transplants it to find point correspondences across consecutive frames of a video.

2. Problem Statement. Methods that rely on being able to distinguish local features will fail on pathological videos where many regions are near identical. These methods try to identify points in a frame by the image patch surrounding them, and thereby ignore significant useful information such as the overall structure of the frame. Our method replaces the local feature descriptors with the global contextual descriptor used by the Belongie, Malik and Puzicha[4] (BMP) shape matching algorithm. This descriptor, here after known as the BMP descriptor, looks at the point's

context within the frame. It can therefore take into account broad structural aspects and work well even on data where local features fail to distinguish points. The properties of the BMP descriptor are desirable in aligning videos where other methods fail, such as low frame rate videos where the content changes significantly between frames. The BMP descriptor relies mainly on a global context rather than pixel intensities, so it avoids making assumptions about the data set maintaining similar intensities throughout. This assumption is inherent in matching descriptors of local features, and also pervades alternative methods such as Lucas Kanade feature tracking[8] and other optical flow methods[6, 5] that try to identify regions moving between frames based on their appearance. The features from the Scale Invariant Feature Transform (SIFT) have also been used for video stabilization[3] due to being invariant under image scaling, rotation, and illumination; however, these features are complicated and expensive to compute, and the SIFT features still rely on the feature tracked reamining visually similar throughout in terms of intensity changes. The BMP descriptor's reliance on contextual information about the points allows it to overcome these other methods in challenging situations where the video frames change not just by changing illumination and movement, but also by changes in the addition and removal of textural information used by these other methods. To our knowledge, the application of the BMP descriptor to video stabilization is novel, although it has already been applied to matching points across images of a scene from multiple angles for the purpose of filling in regions of missing information in one angle using the information seen in the matching region from another angle [7]. We have examined the efficacy of the BMP descriptor for video stabilization and compared this method to a method using local feature descriptors and RAndom SAmple Consensus (RANSAC) to select the best matches[13].

The RANSAC method can be summarized as follows, and it will be described more fully in the comparison with our method in section 5.1. A video frame is passed to a corner detector, which returns a collection of points at locations that appear to be the meeting point of two edges or transitions in the image. Each point is then given a local feature descriptor equal to a small image patch under the point. This process is done on each frame in the video. Then, starting at the first frame, the points in the frame are compared with the points in the next frame by comparing the descriptors of the points. Many potential matches of a point in the earlier frame to a point in the later frame are made. In all but the most trivial of examples, significant portions of these matches will be incorrect and severely hinder the accuracy of the method, so the matches are pared down by RANSAC, which takes random samples of point matches and removes the matches that seem to not agree with the majority of matches. This hopefully results in a set of matches that are consistent in the transformation between frames that they suggest. After this process is performed for each pair of consecutive frames in the video, the matches are used to compute transformations to align each video frame to the preceding one and thereby stabilize the entire video.

This existing method works well with general videos [2], but we have identified two data sets that exhibit properties that lead to poor results and sometimes outright failure. However, the BMP descriptor is well suited to handle even these pathological cases. The particulars of these data sets

will be expanded upon in the following section.

The BMP descriptor is invariant under translation, and robust to small affine transformations[4]; which is applicable because we can reasonably assume that successive frames differ only by a small transformation. However, when the frame rate is low or when trying to align non-successive frames, the misalignment may be too large. We extend the BMP descriptor to be rotationally invariant such that it is still applicable in cases of frame rotation.

FIG. 3.1. *Sea Shell Input Data Set [10]*

3. Input Data. We utilize two types of data in our research, each with its own particular characteristics that prove challenging for local features. We examine contextual point matching for video stabilization using sea shell sketch data and video time lapses of the North Star, Polaris[1].

3.1. Sea Shell Sketch Input Data. We have a sequence of sea shell sketches (Fig.3.1) that were taken at different stages of the drawing. The frames were taken at different camera positions and at different scales. The goal is to stabilize these frames to get a video that, at each frame, shows only the added details to the drawing. In this case, the video stabilization offers a helpful means of creating a smooth time-lapse of a drawing that can help artists view and analyze the process more clearly.

The sea shell sketches data set demonstrates several interesting challenges not found in many videos:

- The early images are done in faint pencil, and the contrast differs sharply from the dark ink present in the later images. This inconsistency forces the method to be highly robust

to changes in aspects other than shape.

- The sea shell displays a high degree of self-similarity due to its spiral shape. Enlarging or rotating the basic spiral shape results in another spiral very similar to the original. This provides a significant hurdle to shape-matching for alignment, and forces our method to take into account all the detail of the sketches to find the aspects which uniquely determine the overall sketch orientation.
- The content of the image sequence changes dramatically. New details are added with every subsequent frame, and the amount added between images varies. This poses a major challenge, as we consider each point's context in the overall point distribution. The distribution changes between frames, but our method is still very robust under such aberrations.

3.2. North Star Input Data. The star data we utilize is a time lapse compilation of stars rotating around the North Star which remains close to the middle through out the video. The background is relatively stable in terms of illumination.

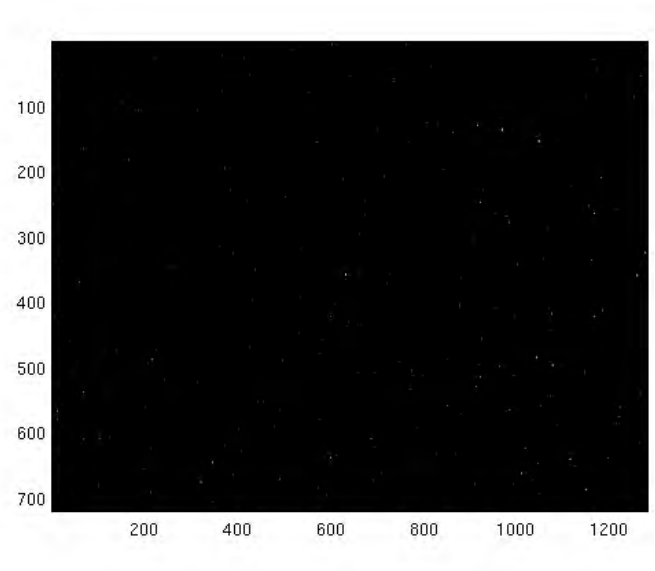


FIG. 3.2. *A frame from star data set*

To align the frames, we need to be able to consistently find stars that match from frame to frame. However, one can observe in the original video that some stars fade in and out in various frames. These flickering stars make the alignment process more difficult. An additional challenge is that local information is insufficient to find corresponding stars; any star-centered image patch is going to look roughly the same as a bright spot on a patch of black. For this data set, it is very important to use contextual information to find correspondences.

4. Algorithm Description.

4.1. Method. Our method is broken into a few main steps. For every frame in the video, the same repeating process is used.

1. Find key points in the image.
2. For each point, compute descriptors that represent important characteristics of that point.
3. For each point, find the key point in the previous frame with the best matching descriptors.
4. Use the resulting matches to compute a transformation to align the two frames.

To find the key points in each frame in the sea shell sketch, we first preprocessed each frame, and then apply an existing corner detector supplied by Rosten and Drummond[11, 12]. We won't go into detail about the corner detector, as the scope of our research is limited to finding point correspondences and other methods of choosing the key points could be used. However, we will further explain the preliminary image processing we did to add robustness to our method later in section 4.2.1. For the star data, we select the centroids of the stars as the key points. The method for doing so is discussed in Section 4.5.

The various descriptors that are computed compose the core of this research, and they are described in greater detail in sections 4.3.1 and 4.3.2. The descriptors capture the useful information for matching points.

The matching is done through a naive, but effective, strategy in which a matching cost is computed for each pair of points across the two frames, and the minimal values are used to find each point's match. More detail on this selection process is given in section 4.4.

The transformation is computed from the point correspondences by solving for a best-fit non-reflective similarity transform. This model finds the best transformation using rotation, scaling, and translation to align the points. These three elements cover almost all expected changes between subsequent frames of a video; large changes in skew, perspective deformations or reflections are unlikely. This model is always over-determined, and we utilize a least-squares optimization to define the best-fit transform.

All the transformations were computed relative to the initial frame by multiplying the sequence of transformation matrices to align the successive frames. This result was more efficient and produced clearer results than sequentially aligning frames to their previous ones. The sequential approach results in significant blurring caused by multiple interpolations.

4.2. Point Selection. Due to the different challenges our data impose, we employed a variety of preprocessing strategies to improve the selection of points in each frame.

4.2.1. Preprocessed Frames for Seashell Sketch Data. Before passing the frame to the corner detector, we apply some basic image processing to help provide consistent detection. The goal is to create frames with much of the extraneous detail and noise removed; the corner detector will produce more consistent points across frames when working with only the major edges within the frame. In the case of our sample data of the seashell sketch, this preprocessing allowed the method to successfully align the early faint pencil sketches along with the later dark ink frames by normalizing the differences in contrast between the sketch and the paper. The exact steps used in

the preprocessing step are as follows, starting with the video frame as a 2D array of pixels with three channels in the Red Green Blue (RGB) colorspace per pixel:

1. Emphasize all the edges in the frame. This is done with a basic edge emphasis filter, with the given kernel below, that assigns each pixel the difference of its neighbors. Thus pixels with wildly different neighbors, such as one on an edge, will get high magnitude values, while pixels in regions of near-constant color will near-zero values. The exact values resulting from this will be later normalized.

$$kernel = \begin{bmatrix} 0 & 10 & 0 \\ -10 & 0 & 10 \\ 0 & -10 & 0 \end{bmatrix}$$

2. Convert the values from the 3-dimensional RGB color space resulting from the first step to simpler 1-dimensional gray-scale values by averaging the intensity of the three channels resulting from step 1 at each pixel.
3. Scale the resulting values to range from 0 to 255 to normalize the resulting data values regardless of the original frame's contrast level.
4. Apply a median filter to remove the numerous specks of noise that have been vastly amplified by the previous steps since edge filters are quite sensitive to noise. Any aberrant specks are removed, while the broad major edges emphasized remain.

4.2.2. Preprocessed Frames for Star Data. Before applying our frame alignment algorithms, we preprocess the star images by removing inconsistent stars that tend to fade in and out. These flickering stars were generally duller and smaller. The brightest stars observed in the video appeared often so we chose to threshold the data for a specific intensity. The challenge then presented to our intensity algorithm was that if we set the threshold too low, we would pick up too many small, dull stars and if we set the threshold high, we would have less data to use. In order to consistently find enough points for comparison between frames, we lowered the threshold adaptively until enough stars were found in the consecutive frame.

In addition to the thresholding challenge, we had to find the centroids of the stars to a high accuracy. Computing the centroid ensured that only the center of the cluster of pixels that made up a star would be matched. Utilizing the corner detection on the star images posed several problems. Each individual star had the possibility of being composed of multiple pixels, which meant that corner detection could pick up on different pixels in various frames for the same star which produced blurry results by leading to inaccurate transforms. Instead we used K-means clustering to find the centroids of the stars. We provided as input to the K-means algorithm a list of all the locations of non-zero intensity pixels and one pixel in each connected component. By using the centroids as the points to align, we reduced the error of our alignment algorithm compared to using the detected corners.

4.3. Computing Descriptors.

4.3.1. Global Contextual Descriptor. The global contextual descriptor is the most fundamental descriptor for our method. It is the novel aspect of our research, and it's properties are relied upon for our method. As mentioned in the previous section, we use a corner detector to select our points for the sketch data. To define the global contextual descriptor, we compute a matrix that represents the relative distribution of the other points in regards to the selected point. Thus the selected point is not characterized by the value of its immediate neighborhood of pixels in the video frame, but rather by its relative position to other points in the overall picture. This global context provides a powerful descriptor for finding correspondences. In the two data sets we are considering, this contextual information leads to more variation in the descriptors relative to local features since many regions in the two videos look locally the same. This increased variation in descriptors allows for significantly better matching by reducing the false matches of locally similar points.

Recall that the contextual descriptor we use is taken from Belongie, Malik and Puzicha[4]. It is based on the notion of creating a 2D matrix that represents a histogram of the relative positions of the other points to encode a point's context in the point distribution. To create this histogram for a given reference point, we create several concentric circles around this point and slice them radially. This creates a set of bins around the reference point, and they are constructed to be uniform in log-polar space per the BMP algorithm. On the left of Figure 4.1 is a visualization of the bins. On the right of Figure 4.1 is the histogram that is generated. Each bin on the circle to the left is mapped to a box on the histogram to the right where each row represents a concentric ring (with the largest at the bottom row) and each column represents an angular slice (starting at 0 degrees in the leftmost column, and proceeding counter-clockwise). The histogram is displayed in gray-scale, in which the bin with the most points is represented as white, and empty bins are black.

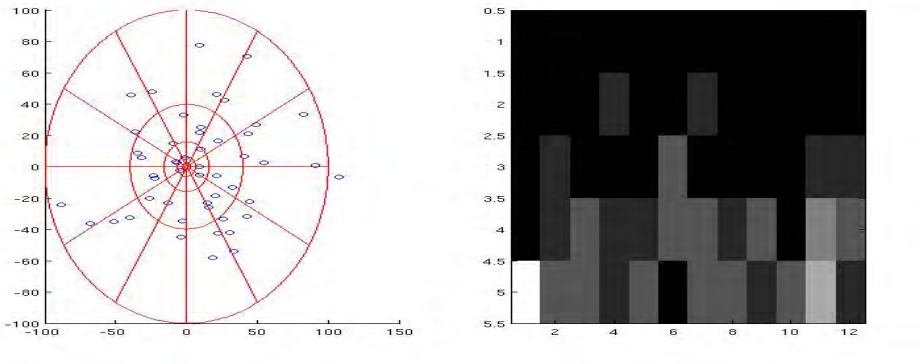


FIG. 4.1. *A Visualization of Contextual Descriptor*

Now that we have a histogram for each point, we need to define a metric to compare two histograms. This metric function given by Belongie et al. [4] is defined for two histograms h_i and h_j as follows:

$$Cost(h_i, h_j) = \sum_k \frac{(h_i(k) - h_j(k))^2}{h_i(k) + h_j(k)} \tag{4.1}$$

where $h_i(k)$ is the value of the k^{th} bin of h_i, along with a similar definition for $h_j(k)$.

The denominator normalizes the bins to keep bins with many points from dominating the comparison. Instead, only relative differences are considered.

Figure 4.2 demonstrates the alignment of two consecutive frames. On the bottom left is the overlay of the original images; on the bottom right is the overlay of the images after the alignment.

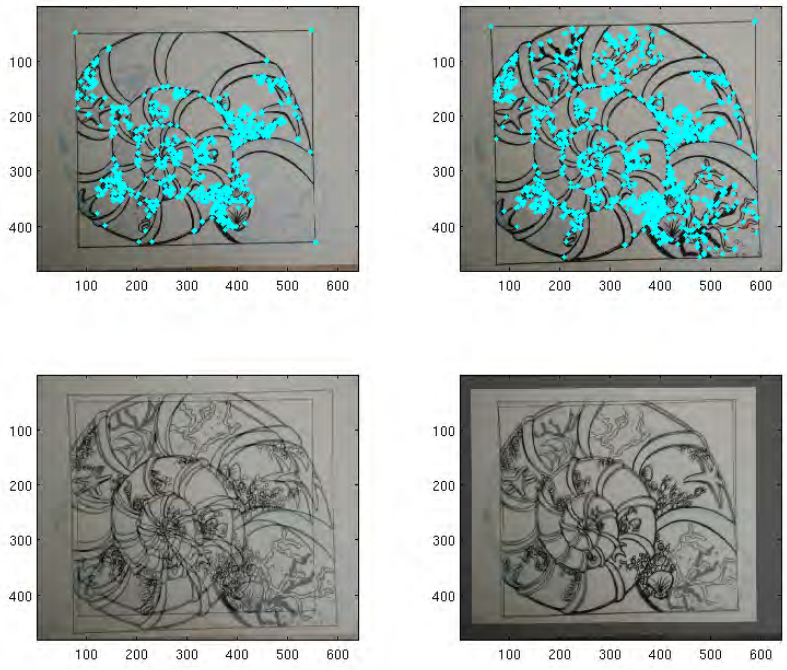


FIG. 4.2. *A Visualization of Two Frame Alignment*

The global contextual descriptor does fairly well when the change between consecutive frames is subtle. However, it is not robust when large amount of detail is added between frames, for instance in the sequence of sea shell sketches, when the light pencil sketch is first replaced by the heavy ink work, many more points are detected as a result of the sharper corners produced by dark ink and light paper. This changes the overall point distribution, and skews the contextual information.

When there are significant rotations between frames, the descriptor fails to produce similar descriptors for the same point since the entire point distribution rotates, moving many points into different angular bins. This failure of the descriptor leads to incorrect matches which results in a miscalculation of the transformation. Later, we will show how to make the contextual descriptor rotation invariant in section 4.3.5.

4.3.2. Adding in Local Descriptors. Our method also uses some additional local features to create an even more robust system of describing points. We use two such descriptors to augment the point matching. One uses color, which imitates the existing method of using image patches as local features, except that it uses only a single pixel per point rather than a patch made of several patches. This color descriptor is quick and easy to compute, and offers information not gathered in the BMP descriptor. The second extra descriptor uses the radial histogram of the BMP descriptor to count up nearby pixels of high intensity edge transitions. It is in effect describing the local structure of edges near the point, which offers additional information not found in our primary BMP descriptor, which describes the structure of the distribution of all the points selected for the current frame.

By also considering other descriptors, the matching can be improved. Though the BMP descriptor works fairly well on its own, multiple independent sources of information give the algorithm better chances at finding good point matches. For each additional descriptor, a new metric is applied between every pair of points resulting in more costs that quantify the difference between two points. A linear combination of the metrics for each descriptor then results in an overall metric, as shown in equation 4.2, that synthesizes the various information on point similarity from the different descriptors and metrics. This synthesis of independent metrics adds robustness by letting the different metrics balance out flaws in the other metrics.

Other local features may be useful, and this leaves room for future work. Adding more independent sources of information tends to lead only to improvements.

The equation used for synthesizing the three metrics (BMP, local color, local edge) is given below. $Cost_i$ represents the cost computed for "difference" between the two points p_1 and p_2 being compared according to the i^{th} metric. The coefficients, w_1 ... w_n, for this linear combination were experimentally optimized.

$$Cost_{total}(p_1, p_2) = \sum_{i=1}^{n} w_i Cost_i(p_1, p_2) \tag{4.2}$$

$Cost_{total}$ contains algorithms perceived difference between two given points, which will be used in finding point correspondences. If $Cost_{total}(p_1, p_2)$ is low, p_1 and p_2 are points that are supposedly matching.

4.3.3. Local Descriptor: Color. The color descriptor utilizes the color of the point in the image. It depends only on the point itself, not the overall point distribution. Thus, it is independent of the global contextual descriptor. This descriptor is useful, but it is not too effective alone. This is particularly true in our case, where the image is almost entirely shades of white, gray, and black. Many different points in an image will have the same RGB values; this cannot provide unique matches. Furthermore, many points that should be matched will have different RGB values due to changes in illumination and noise from the camera sensor. However, since the different metrics are weighted accordingly, this information helps without hindering.

At each selected point, we use the RGB values of the single pixel at each point to compute the color cost according an Euclidean distance metric:

$$Cost(\vec{r_i}, \vec{r_j}) = (\frac{\vec{r_i}}{||\vec{r_i}||} - \frac{\vec{r_j}}{||\vec{r_j}||})^2 \tag{4.3}$$

The subscripts i and j represent the two frames we want to align. $\vec{r_i}$ and $\vec{r_j}$ are the three-dimensional RGB vectors of the pixels under the two points to be compared with the red, green, and blue values at the points as independent components. Normalization adds robustness to illumination changes; if one frame is dimmer than the other, this change will be cancelled out.

4.3.4. Local Descriptor: Edge. Another local descriptor we used is the local edge information. This local descriptor is similar to the BMP descriptor since it also utilizes the contextual radial histogram bins of the BMP descriptor. However, it differs from the BMP descriptor in two ways. First, this one uses the distribution of high-intensity edge points rather than the distribution of the other reference points (which were found as the corners or centroids in the frame). Second, this descriptor is restricted to a localized area around the point; the radius of the farthest bin in the histogram is significantly smaller than in the global context histogram. Only the local edge features of the image are considered.

The process to compute this descriptor for a given reference point is as follows:

1. Apply a Prewitt filter, which highlights the edges of the image, to separate out a good set points:
$$Prewitt filter = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$
2. In the resulting set of points, select the points with the maximum intensity to form the high-intensity edge data point set.
3. Compute a log-polar histogram of the distribution of high-intensity edge points nearby the reference point, as done for the BMP descriptor.

Figure 4.3 shows the result of applying the Prewitt operator to an early stage of the sea shell sketch. On the left is the original light blue sketch that is very faint. On the right is the same image after we apply the filter. The filter separates out a clear set of high-intensity edge points that we can use.

To compute the cost for comparing these two metrics, we apply the same histogram metric (Eq. 4.1) as used in comparing the histograms of the BMP descriptor.

4.3.5. Rotationally Invariant Descriptor. If we apply the frame alignment algorithm presented up to this point on two frames of star data that are significantly rotated, we obtain poor alignment. This is shown in Figure 4.4 and motivates the need for rotational invariance in our descriptor. The lines in the figure indicate point matches between two frames; the lines connect a point in the frame to the coordinates of the best-matched point in the subsequent frame. This

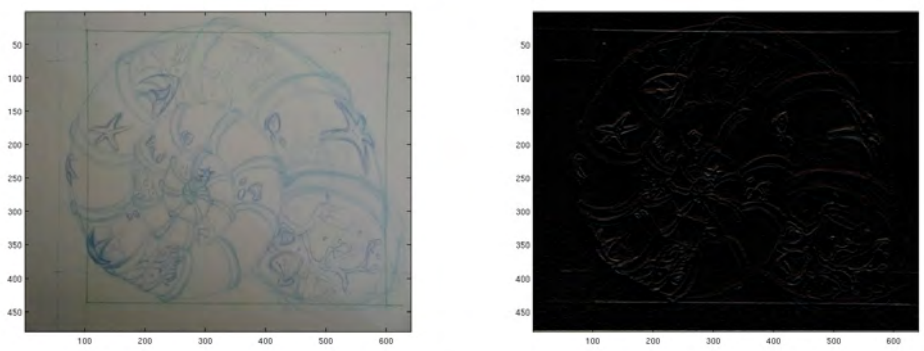


FIG. 4.3. *Applying Edge Filter*

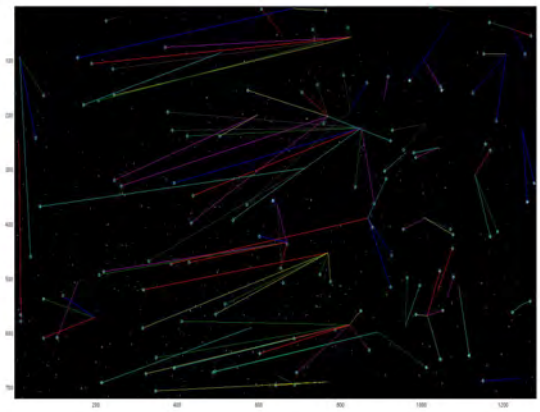


FIG. 4.4. *Frame 1 and Frame 1000 alignment and point correspondences*

figure shows that these results of the point correspondences are far from what we expect since there are many long lines showing no consistent motion for the stars as a whole. Therefore the set of descriptors described above has shortcomings when it comes to frames that are rotated.

To handle significantly rotated frames, we modified our descriptor by using the Discrete Fourier Transform (DFT) to make it rotationally invariant. The DFT can be interpreted as a change of basis in a discrete setting. The coefficients are related to the frequency content of the signal as defined by equation 4.4. The magnitude is unchanged by periodic shifts, which is useful because the rotation of points is a periodic shift of the rows in the histogram which represent the angular bins of a particular concentric circle around the point. Therefore we can extend our contextual descriptor to be rotationally invariant by computing the DFT of the rows in the histogram and keeping only the magnitudes. This descriptor is a better alternative than using the normal aligning frames algorithm because it neutralizes the periodic shifts that one can witness in the context histogram. Whereas the normal aligning frames algorithm would not be able to recognize the point correspondences between rotated frames, our rotationally invariant aligning frames algorithm is able to match the

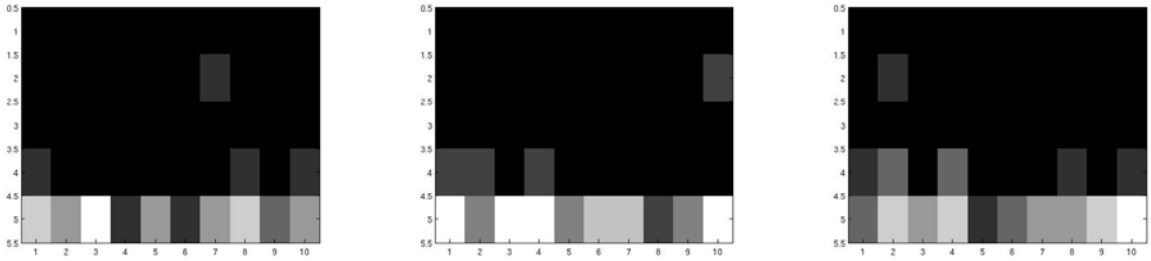
frames with the same given information. However, information is lost when taking the magnitude of the DFT coefficients, so when rotation is not an issue, the non-rotationally-invariant descriptor may be more informative. The k^{th} DFT coefficient of a vector x is defined by

$$X_k = \sum_{n=0}^{N-1} e^{(-2\pi ikn)/N} x_n, k = 0, 1, ..., N-1 \ . \tag{4.4}$$

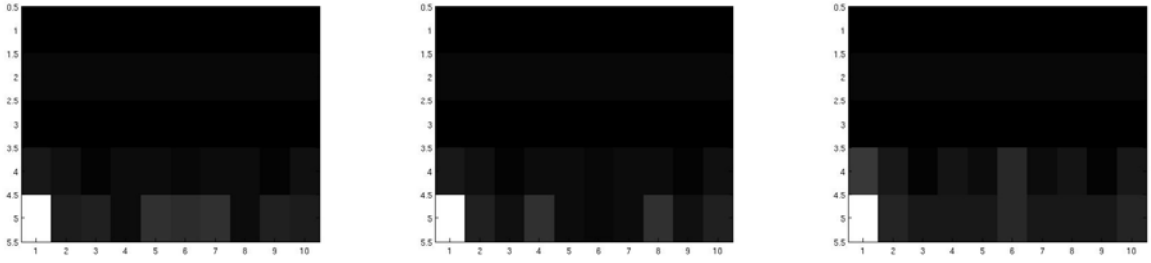
A shift of s in the input vector x, causes X_k to be multiplied by a complex number with an absolute value of 1. The following demonstrates why the magnitudes of the DFT coefficients are independent of periodic shifts.

$$X_k e^{(-2\pi isk)/N} = \sum_{n=0}^{N-1} e^{(-2\pi ikn)/N} x_{n-s}, k = 0, 1, ..., N-1 \tag{4.5}$$

$$|e^{-2\pi isk/N}| = 1 \qquad \text{so,} \qquad |X_k e^{-2\pi isk/N}| = |X_k| \ . \tag{4.6}$$



(a) Histogram of frame 1 (b) Histogram of rotated frame 1 (c) Histogram of frame 1001



(d) Rotationally invariant histogram of frame 1 (e) Rotationally invariant histogram of frame 1 rotated 90 degrees (f) Rotationally invariant histogram of frame 1001

One may notice that each histogram in the top row appears to have adopted a shift, in which some of the boxes have reappeared on the other side. Frame 1 and the artificially rotated Frame 1 are clearly different histograms; many bins have been altered significantly. With our rotational invariance descriptor applied to these two frames, the histograms appear very similar in comparison. Below, we demonstrate the effect of our rotationally invariant descriptor. With the Fast Fourier Transform applied directly to it, our context histograms display a very close match. Although there are some dissimilarities in the middle of the bottom most row, we must account for the fact that

in Frame 1001, there are markedly less stars of the threshold intensity in comparison to Frame 1 where there are many more. In Frame 1001, we had to raise the intensity threshold slightly to find at least as many stars in Frame 1. Another reason the rotationally invariant histograms were not identical was rotations of the image lead to only approximate periodic shifts in the original histogram since the rotations don't guarantee all points in a bin will end up together in the same bin after the rotation. With that in mind, the rotationally invariant histograms demonstrate great similarity.

4.4. Matching and Aligning. For every point selected in one frame, we find its best match in the next frame according to our overall metric (equation 4.2). Then we use least squares optimization to calculate a non-reflective similarity transform given the input matches as shown in the following equation (4.7). x_i^{in} and y_n^{in} represent the coordinates of the i^{th} point of the frame to be transformed, x_i^{out} and y_i^{out} are the coordinates of the point matching (x_i^{in},y_i^{in}). sc, ss, t_x and t_y are the 4 parameters of the transformation we are solving for that best aligns the *in* points with their corresponding *out* points.

$$\begin{bmatrix} x_1^{out} & y_1^{out} \\ \vdots & \vdots \\ x_n^{out} & y_n^{out} \end{bmatrix} = \begin{bmatrix} x_1^{in} & y_1^{in} & 1 \\ \vdots & \vdots & \vdots \\ x_n^{in} & y_n^{in} & 1 \end{bmatrix} \begin{bmatrix} sc & -ss \\ ss & sc \\ t_x & t_y \end{bmatrix} \tag{4.7}$$

4.4.1. Point Matching. To compute that transformation, we need to find pairs of points to match across the frames. One naive way for matching is to consider each point in one frame, and find the point in the next frame with the smallest value from our metric. In other words, for a given point p_j in frame n, find point p_k in frame $n+1$ that minimizes $Cost_{total}(p_i, p_k)$. This selection process is simple, and fairly effective. However, it does lead to a high amount of shared matches, where two or more points in one frame map to a single point in the next frame. In cases where the earlier frame has more points, this is guaranteed to happen.

These duplicate matches are particularly bad. Consider two points in one frame being mapped to a single point in the second frame, where one match is correct and the other is wrong. Our transformation calculation uses a least-squares optimization, which will try to appease both matches, and thus will fully realize neither. A good match is almost completely neutralized by a bad shared match. In cases where a point has no true matching point in the other frame, it is very likely to match up with a completely unrelated point and lead to unsatisfactory transformations.

Instead, we apply a simple greedy algorithm, a short-sighted algorithm that always picks whatever choice looks best at current moment, to find the best one-to-one matches. This process is simple and quick, but not guaranteed to produce the optimal set of point correspondences. However, it has the nice property of protecting the best matches from the influence of the others, since each point only get matched once. Points with no good match will be saved for later, and at worst will get matched with other bad points.

The algorithm can be expressed as:

1. Find the pair of points between the frames that are closest as defined by our metric.
2. Declare those two points a match and remove them from future consideration.
3. Repeat the above steps until we run out of points in either frame.

4.4.2. Eliminate Bad Matches. The greedy algorithm works, but it still leaves some points poorly matched, especially since it continues until one frame has run out of points despite the presence of points that have no valid match. We try to remove the worst matches to improve our estimate of the transformation parameters. We go about this by removing the outliers of the transformation. This process has four main steps:

1. Use all the matches to compute a transformation, as laid out in equation (4.7).
2. Apply the transformation to the reference points in one frame and compare their positions to their presumed matches in the other frame.
3. Throw away the 20% of matches with the farthest distance after the transformation.
4. Recompute the transformation with the remaining matches.

If the point matches were ideal, the transformation would be exact and the transformed points would line up exactly with their corresponding points. Assuming the transformation is reasonably good, points will at least end up near their corresponding points. Points which remain far from their corresponding points after a transformation are likely incorrectly matched. Under our testing, the initial transformation from the greedy algorithm's matchings is good enough for this to be reliable; the inaccurate transformation should still map the correct matches significantly closer than the invalid matches. Just running one pass of this refinement algorithm provides effective results in removing bad matches. The final transformation is computed using just the remaining matches, once again using equation (4.7).

5. Results.

5.1. Comparison to RANSAC: Sea Shell Sketch. As a test of our method, we compare it to a prominent video stabilization method using local features to find point matches and RANSAC [13] to find the best transformation. The local features provide numerous point matches, many of which are incorrect. The basic idea of RANSAC is to pare down the matches by taking random samples of the point correspondences and finding inliers. Point correspondences that do not appear to fit with the same transformation as most of the group are labeled outliers and removed. This provides good results when given point correspondences that are mostly correct, but it relies on random chance. The non-deterministic nature of this method can be undesirable. One cannot always depend on RANSAC to give the same exact result each time.

Our method finds much more meaningful point matches. The ratio of correct matches to incorrect matches is high enough that reliable frame alignment can be achieved without needing to resort to the probabilistic techniques of RANSAC. This method is completely deterministic; the same input data and parameters will always return the same alignment. To compare the two methods, we provided both the same preprocessed Sea Shell Sketch frames and examined

their respective outputs. This was done with a pre-existing implementation of RANSAC video stabilization [2].

The comparison yields some promising results:

- RANSAC's stabilized video result isn't consistent. For every run, we get a different alignment of the video.
- Most of the runs of RANSAC failed significantly. Incorrect transformations were common, often translating the image completely out of the frame or zooming in on a small portion of the sketch.
- For RANSAC, although the last few frames done in ink were often aligned correctly, the alignment of the first few frames done in the light blue pencil sketch failed completely. They were not aligned at all.

This example data shows a case in which the existing method relying on local features fails, for the video frames contain many regions that appear locally to be the same. Furthermore, the self-similarity of the spiral lead to incorrect scaling transformation being supported by many of the matches, which lead to unreliability in the random samples of RANSAC. Our method using the BMP descriptor and its use of contextual information allows it to succeed even on this particularly troublesome data set.

FIG. 5.1. *Comparison of Average Frames*

The results of our video stabilization method are very encouraging. In Figure 5.1, we show the average frames comparison for the sea shell sketch. On the left of Figure 5.1 is the average frames of the original sketch sequence; on the right, is the average frames of the aligned sketch sequence. Although slightly blurry, most of the shifts and rotations have been eliminated from the aligned frames, leading to a crisp average of the frames.

5.2. Star Alignment Result. Recall that the comparisons made with synthetic data were shown in Section 4.3.5. Here we use real data and try to align frame 1 and frame 100.

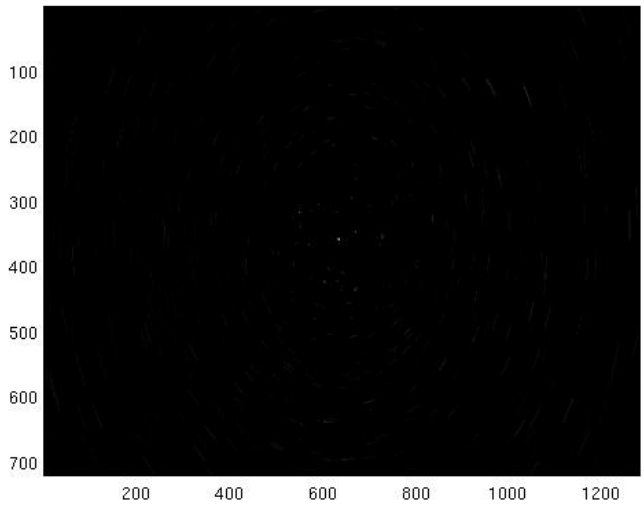


FIG. 5.2. *Frame 1 to frame 100 unaligned and averaged onto one image*

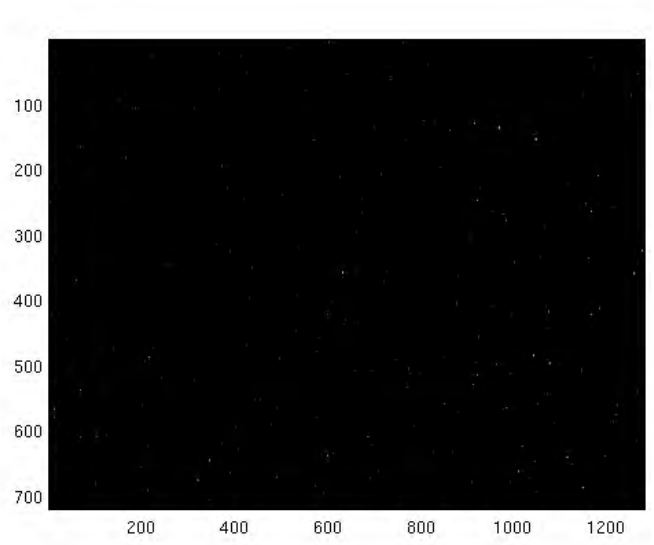


FIG. 5.3. *Frame 1 to frame 100 aligned and averaged onto one image.*

The figure 5.2 shows the sum of the first 100 unaligned frames from video, and figure 5.2 shows the sum of the first 100 aligned frames, which demonstrates the capabilities of this frame alignment algorithm. In the unaligned frames, the stars are visibily smeared as they rotate out of place; in the aligned frames, each star appears to a crisp stationary point of light.

6. Conclusion and Future Work. As shown through our method, use of contextual information is beneficial in video stabilization. As we demonstrate through our specific data sets, utilizing the contextual information can help increase the robustness of video alignment, especially when used on videos with properties that may confuse other algorithms. While we need further in-

vestigation to determine how widely applicable this method is, we have successfully demonstrated that this technique of using contextual information is viable for video stabilization, even when confronted with some pathological data sets.

There are directions left for future work. First, although this method proves to be less computationally expensive than we originally thought, it still takes quite some time, especially when the amount of points selected increases drastically. Therefore looking into reducing the runtime cost is definitely a good idea. Second, our method as it stands currently reduces almost all the camera movement; however, there are still very small shifts left between frames. To eliminate this remaining inaccuracy and have a perfect alignment, we need to look further into how to find matches. One avenue would be to integrate these contextual descriptors into other existing methods. Other potential improvements include considering more than two frames at a time to create smoother and more consistent transformations, and exploring new means of finding key points in each frame. Trying to cluster points based on groups which appear to move together can lead to more flexibility and a possible tracking application. For example, in a video containing both a moving foreground and a moving background, being able to choose which movements are stabilized could be useful.

Furthermore, we want to try our method on other data sets and test its robustness on videos that have other special properties that challenge the conventional stabilization approaches. By testing our method on those videos, we hope to further improve our method and eventually develop a robust algorithm that achieves perfect alignment on numerous types of videos without requiring intervention and fine-tuning by the user. We also would like to compare our descriptors to more existing methods.

REFERENCES

[1] Time lapse north star, http://www.youtube.com/watch?v=XiaSKtkjC9M.

[2] Video stabilization using point feature matching. 'http://www.mathworks.com/products/computer-vision/examples.html?file=/products/demos/shipping/vision/videostabilize_pm.html'.

[3] Sebastiano Battiato, Giovanni Gallo, Giovanni Puglisi, and Salvatore Scellato. Sift features tracking for video stabilization. In *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*, pages 825–830. IEEE, 2007.

[4] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(24):509–522, April 2002.

[5] Hung-Chang Chang, Shang-Hong Lai, and Kuang-Rong Lu. A robust and efficient video stabilization algorithm. In *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on*, volume 1, pages 29–32. IEEE, 2004.

[6] Jyh-Yeong Chang, Wen-Feng Hu, Mu-Huo Cheng, and Bo-Sen Chang. Digital image translational and rotational motion stabilization using optical flow technique. *Consumer Electronics, IEEE Transactions on*, 48(1):108–115, 2002.

[7] Sung Ha Kang, Tony F. Chan, and Stefano Soatto. Inpainting from multiple views. In *First International Symposium on 3D Data Processing, Visualization and Transmission*, pages 622 – 625, 2002. Conference paper.

[8] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

[9] S. Peleg M. Ben-Ezra and M. Werman. A real-time video stabilizer based on linear programming. 1999.

[10] Ling Han Meng. Sea shell, 2012.

[11] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, volume 2, pages 1508–1511, October 2005.

[12] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision (to appear)*, May 2006.

[13] Chunhe Song, Hai Zhao, Wei Jing, and Hongbo Zhu. Robust video stabilization based on particle filtering with weighted feature points. *IEEE Transactions on Consumer Electronics*, 58(2):570–577, May 2012.