Automatic Generation of Action Sequence Images from Burst Shots

Sean Chen, Ben Stabler, and Andrew Stanley
Emails: ssch, stabler, and astan @stanford.edu
EE 368: Digital Image Processing, Stanford University

***Abstract*—Many sports enthusiasts, from novice photographers to professional publishers, rely on manual image segmentation with tools like Photoshop to combine multiple images of a bike trick or basketball dunk into a single image by cutting out the foreground of each image and overlaying it onto the background of one image. The goal of this project is to develop an algorithm that can automatically combine multiple images generated from burst shots of an action into a single image that clearly shows the full action. This requires three main tasks for each set of images, including background alignment between images in the cases when the camera is moving (using feature detection and matching), segmentation of the foreground and background components of each image even in cases when portions of the background might be moving, and finally cleanly combining the foreground image segments all of the images onto a single background image. The algorithm described in this paper successfully compiles a variety of image sets, including those where the foreground object overlaps between images or sets with multiple objects, but fails to compile sets where multiple objects cross paths during the action.**

June 5th, 2013

I. Introduction

To implement an algorithm that can successfully stitch together the foregrounds from an action sequence onto a single background requires the incorporation of a broad spectrum of image processing and computer vision techniques. Many of these techniques, from feature detection and image filtering to segmentation and point operations have been researched in depth, providing a multitude of options as starting points for each branch of this project. While panorama "apps" are now ubiquitous on mobile devices and often even come preloaded on smart phones, the research for creating full view panoramic image mosaics goes back to the 1990's [7]. This work, in combination with more recent work like speeded up robust features (SURF) [1], provides a solid base of knowledge from which to work for the image alignment necessary to do clean point operations between images in a set.

Many alternate applications provide a base for background removal as well. For example a security company interested in automatically detecting intruders from a video feed may drive development of background subtraction algorithms. One research group has successfully used optical flow to perform background subtraction [4], and [6] even provids a full review article of the state-of-the-art for background removal in 2008. Drawing on previous research and the multitude of techniques covered in coursework, the algorithm presented in this paper aims to optimize similar image processing techniques to automatically create action sequences from burst shot photos.

Fig. 1. An example of an action shot sequence created manually using a program like Photoshop. Our algorithm aims to automate the generation of such images. Source: hongkiat.com

II. Algorithm Implementation

A. Image Alignment

The frontend phase of our approach involves an image alignment step to account for the fact that unless the camera is perfectly stationary on a tripod, there will be some angular and translational motion between successive images within a burst. Since the rest of our pipeline relies on the images taken to be roughly aligned, it is necessary that we first calculate the homography between various images in the burst sequence and then apply perspective transforms to ensure alignment. Specifically, we consider the very first image within a burst to be the reference perspective, and we apply transformations on all successive images within the burst to match this reference perspective.

There are several types of feature detection approaches, including MOPS, SIFT, and SURF. SURF is an efficient feature detector and is invariant to multiple types of image transformations, including rotation and scale, and so in our approach, we use SURF to detect our features because we have experimentally observed it to give the fastest results with no discernible loss in accuracy for our test images. The result of applying SURF on our images gives us a set of feature

Fig. 2. The first and last image of the set used as an example in Section II-B.

descriptions for each image, as well as a series of keypoint locations.

Once we have set of feature descriptors and keypoints for each image, we can proceed to match the features in each successive image within the burst to the those in the reference image. Here, we use the Fast Approximate Nearest Neighbor Search Library (FLANN) [5] to perform a fast and rather accurate list of matches between sets of features. Within this list of matches, we calculate the minimum distance between pairs of features, and we apply a threshold such that we only consider good pairs to be those with a maximum of double the minimum distance.

Having a set of matched feature pairs, where each feature has a keypoint location, we can then calculate the 3x3 homography matrix between each burst image and the reference. We use the RANdom Sample Consensus (RANSAC) [2] algorithm here to calculate this affine model and then apply the perspective transformation. Note that once we warp the image to match that of a reference, there will necessarily be some regions of the newly aligned image that only have black pixels, since camera movement means that certain parts of the reference scene would not be captured in successive images. We simply paste back the original pixels from the reference image to fill in the gaps here.

Image alignment using this method results in nearly perfect alignment for small camera movements. If the camera moves too much between burst images. the quality of the alignment becomes much poorer, where often only a small portion of the image is aligned, and the rest of the image encounters large discontinuities in comparison to the reference. This is especially apparent when there is a great amount of translational motion. This is because when the scene is not planar (e.g. not all objects are relatively far from the camera), homography only holds in cases where the camera is purely rotated.

B. Background Extraction

Aligning all of the images makes it possible to apply point operations to extract the background from an image set. The current algorithm relies on only the first and last image to extract the background under the assumption that those two images will have the least amount of overlap between respective foreground objects, but this could be adjusted to include all images in a set to improve results. Figure 2 shows these two images for the set that will be used as an example in the remainder of this section. The one user input this function requires is the number of foreground objects to keep. For example in this image the number of objects is set to one to avoid including the people walking in the background in the final image. Setting the number of objects higher is preferable for many sports images like the football catch shown in Figure 12.

The first step of the background extraction is to convert both images from RGB to HSV space and then to compute the absolute value of the difference between the two images at each pixel location. Converting to HSV before taking the difference makes the algorithm more robust to changes in exposure between images because, whereas all three channels of an RGB image are affected by a change in exposure, the hue and saturation channels of HSV are unaffected by exposure changes. Next the hue channel difference needs to be adjusted to account for the fact that hue is a circular spectrum, so very high and very low values of hue are actually very similar to each other. Figure 3 demonstrates the effect of accounting for this fact when taking the differences between hue channels, which reduces some of the background noise. The difference between the two images in HSV is converted to a grayscale image by treating each pixels as a vector with three elements and taking the norm of this vector.

Next the grayscale difference image is thresholded to convert it to a binary mask. Selecting a threshold that works

Fig. 3. The effect of accounting for the circularity of the hue channel before taking the difference in hue between images. Noisy pixels on the ledge that seem very different in the left image are actually very close together in the hue spectrum because a hue value of 1 is equal to a hue value of 0. The right image accounts for this fact when taking the difference between the hue channels of two images.

Fig. 4. The result of thresholding the difference of the images in HSV to create a binary mask.

Fig. 5. The result of opening the thresholded image with a disk structuring element.

well with all image sets proved to be one of the most challenging aspects of designing this algorithm. Otsu's method for unsupervised thresholding tended to provide a threshold value that was too high and resulted in much of the foreground object getting cut out of the mask, even when used with a two level threshold with hysteresis as described in [3]. The median and mean pixel value did not provide consistent results between image sets. Ultimately this threshold was set to the value that had the largest jump to the next value when excluding the top and bottom one percent of pixel values. The result of this thresholding is shown in Figure 4.

While the people moving in these image certainly stand out in the thresholded difference, there is also a lot of undesirable background noise, mostly from the trees moving in the wind. To help break up this background noise into smaller chunks that can more easily be filtered out, morphological image processing is used to open the image. This consists of an erosion followed by a dilation using the same structuring element. In this case, a disk of radius two was chosen as the structuring element based on results with several sets of test images. The result of this opening operation is shown in Figure 5.

Fig. 6. The mask with only the n largest regions and all holes filled.

The regions of this mask were labeled using the 8-pixel neighborhood version of the region labeling algorithm and then sorted based on their areas. Only the largest n regions were kept for background extraction, where n equaled four times the input number of foreground objects plus one. As shown in Figure 6, any holes in these regions were filled to provide a cleaner mask.

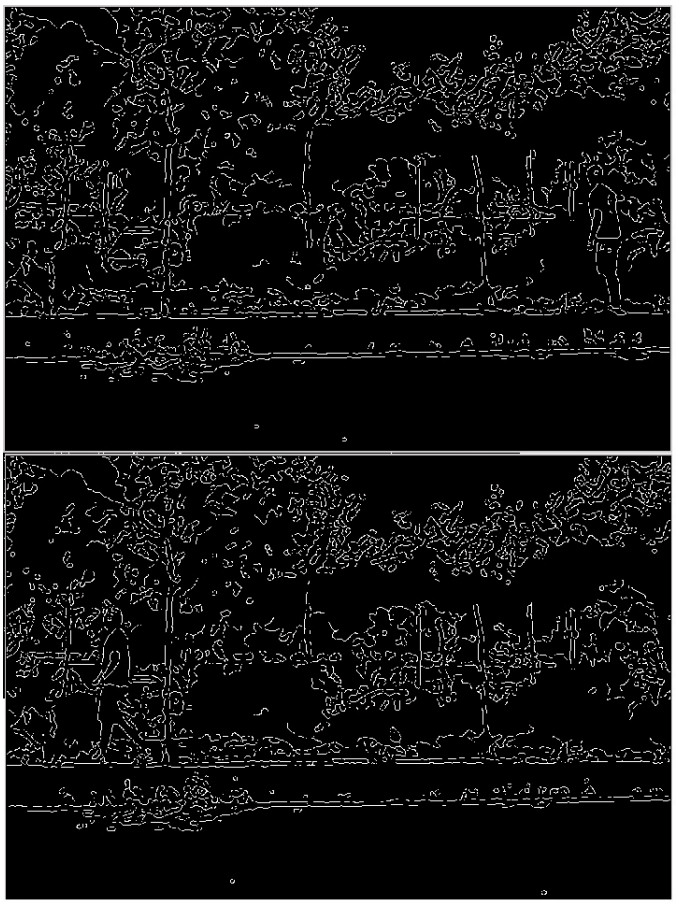

Fig. 7. A Canny edge-detector applied to the hue channel of each of the images provided the cleanest results for comparing to the edges of the regions in the mask based on results from several test sets of images.

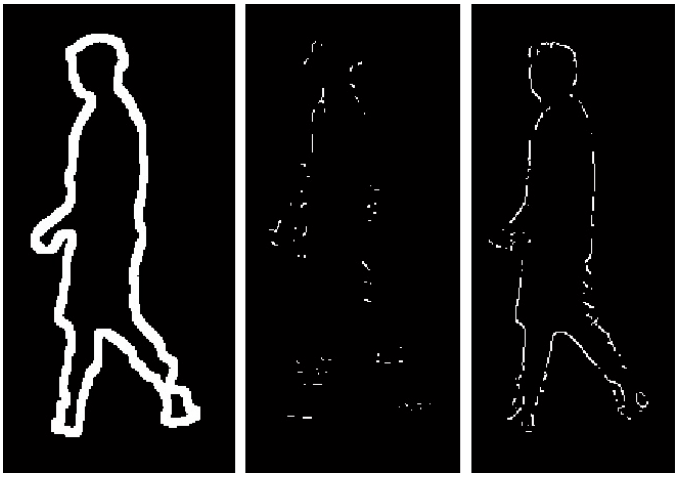

Fig. 8. The dilated edge of one of the regions from the filtered difference mask is shown on the left. The middle and right images are calculated by an AND operator between the left image and each of the hue channel edge images shown in Figure 7. Since the region correlates significantly better with the second image, the first image is used as the background for this region.

Fig. 9. The background calculated from the first and last image in the sequence used to compare against each image individually to pick out foregrounds to compile. All of the moving objects have been successfully removed without any major noticeable artifacts.

Finally, each region needed to be classified as an object of the first image or an object of the last image before the background could be extracted from the two images. This was implemented with an edge-detection based approach. First, a Canny edge detector was applied to the hue channels of each of the images with σ set to three based upon experimentation with several image sets, the results of which are shown in Figure 7. For each of the regions in the binary mask shown in Figure 6, a new pair of region masks were created by an AND operator of the dilated edge of the specific region with each of the two full edge images shown in Figure 7. Figure 8 shows the dilated edge of the region on the left side of the mask and these combined edges for both images.

A correlation coefficient is calculated for the combined edges as the number of white pixels in each combination divided by the total number of white pixels in the dilated region edge. If the two correlation coefficients differ significantly, as they do in the example shown in Figure 8, then the region

Fig. 10. The final image with all foregrounds compiled onto the background.

with the lower coefficient is selected as the background and the region with the higher coefficient is selected as the foreground. The background segments are pasted together on the first image to create the background shown in Figure 9.

Situations where the combined edges did not correlate significantly better with one image than the often indicated an overlap between the foreground in the images. This special case required a bit more work before selecting background elements to compile. To pick out only sections of the region that correlated better with the second image, the combined edges of the first image are closed with a disk structuring element of radius twenty-five and then subtracted from the combined edges of the second image dilated with the same structuring element. The portion of this mask that matches the original region mask is selected as the foreground of the second image that is different from the first and the remainder is chosen as the background. An example result for one of these situations is shown in Figure 11 which, while far from perfect, provides a much better result than simply selecting the image with the slightly higher correlation coefficient for each region.

Fig. 11. Image sets with overlapping foregrounds in all images create a greater challenge for foreground segmentation that has not been perfectly achieved by this algorithm.

Fig. 12. Running the algorithm with the number of objects set to two rather than one prevents the football from getting cut out while it is still in the air.

C. Foreground Masks and Image Compiling

Once the background has been calculated, an algorithm almost identical to the process described in Section II-B is used to extract the foreground from each image in the set. The main difference is when keeping the n largest regions from the difference mask, n is set to be equal to the input number of desired foreground objects. In addition, before adding the new foreground on top of the existing background it is dilated with a structuring element of radius fifty to clean up any pieces that the algorithm may have missed. However, to prevent this dilation from cutting off any of the previously pasted foregrounds, the algorithm keeps track of the pre-dilation masks from all the previous foregrounds and does not dilate the new mask beyond these boundaries. Figure 13 shows some additional examples of compiled images sets created using this algorithm.

III. Conclusions and Future Work

This project shows that quality action sequence images can be generated automatically from a series of burst shot photographs. The algorithm can handle objects that are partially occluded by the same object in subsequent frames as well as image sets in which keeping multiple objects is desirable, like a football being caught out of the air by a running subject.

Despite the successes of this project, the algorithm remains far from perfect and, beyond tuning the various thresholds to improve the current algorithm, other more fundamental issues provide compelling motivation for future work. The most obvious challenge comes from situations in which multiple foreground objects are desired and they overlap each other in different frames. In its current implementation, the algorithm has no way to discern which object was closer to the camera in its respective frame, so the object from the later frame will be pasted on top by default, regardless of whether it is in front of or behind the other object. This can subsequently also throw off other aspects of the algorithm as shown in Figure 14.

Ideally this algorithm could become a compelling application on a mobile device, but its current implementation takes over a minute per image when ported to a mobile device, compared to a couple of seconds on a laptop. One of the bottlenecks here is clearly the image alignment (which is entirely required for a mobile camera environment), but this is difficult to sidestep as we are already using SURF, FLANN, and RANSAC, which are generally the fasted algorithms in this arena. We have tried some experiments with running the

Fig. 13. More examples of images compiled with this algorithm.

Fig. 14. The algorithm fails when trying to keep two objects that overlap because its current implementation has no way of finding which object is closest to the camera.

feature detection, matching, and homography calculations on downsampled versions of the actual images, but this ends up giving much more alignment errors, where regions of the successive burst images become poorly aligned to the reference. An alternative we could later explore, if given enough time, is to try having all the calculations done on a backend server, where the mobile device is merely a thin client that capturs, sends, and receives images.

In addition to optimizing for speed, it would be useful to add features to make the application more user-friendly to lay people to allow them to tune various aspects for their specific image sets, such as transparency of foregrounds.

Breakdown Of Work

The overall breakdown of work was essentially equal between team members. Sean wrote the code for the image alignment within the OpenCV framework. Andrew prototyped most of the image segmentation and compilation algorithms in MATLAB with some help from Ben. Ben then wrote the OpenCV code to implement the MATLAB prototypes and integrated this code with Sean's code. Sean ported the algorithm to run on a iOS device, but the performance was too slow for a good demonstration. Ben created the poster with some help from Sean. Andrew wrote most of the report with a section and edits and other additions from Sean.

References

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. *Computer VisionECCV 2006*, 2006.

[2] Martin Fischler and Robert Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Comm. of the ACM 24*, pages (6): 381–395, 1981.

[3] Gilad Halevy and Daphna Weinshall. Motion of disturbances: detection and tracking of multi-body non-rigid motion. *Machine Vision and Applications*, 11(3):122–137, March 1999.

[4] Wei Li, Xiaojuan Wu, Koichi Matsumoto, and Hua-An Zhao. Foreground detection based on optical flow and background subtract. In *2010 International Conference on Communications, Circuits and Systems (ICCCAS)*, number 1, pages 359–362. IEEE, July 2010.

[5] Marius Muja. FLANN, Fast Library for Approximate Nearest Neighbor. http://mloss.org/software/view/143. 2001.

[6] Khaled M. El-Sayed Shireen Y. Elhabian and Sumaya H. Ahmed. Moving object detection in spatial domain using background removal techniques-state-of-art. *Recent Patents on Computer Science*, (2):32–54, 2008.

[7] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97*, pages 251–258, New York, New York, USA, 1997. ACM Press.