Extensions to Low-Noise Viewfinder Alignment Algorithm

David Chin-lung Fong*
iCME, Stanford University

Shiwei Song†
CS, Stanford University

Abstract

Digital viewfinders play an important role in the user interface for modern digital cameras, but under low-light conditions, the viewfinder images can be very noisy. One way to reduce noise is to take successive frames, perform pixel-level alignment, and take the average of each corresponding pixel across several frames. There has been previous work [Adams et al. 2008] for viewfinder alignment to reduce noise. We extend the previous alignment algorithm to be more robust under object motion and rotation. A difference mask is used to perform smart averaging to handle object motion, while a subregion alignment algorithm is designed to deal with rotations where the original algorithm would fail.

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

Keywords: mobile, visual computing, computational photography, enhancement

1 Introduction

Nearly all cell phones available today are camera phones, but they still suffer from poor performance in low-light environments and low dynamic range. The continually improving performance of phone hardware and budding availability of programmable development environments for mobile devices have enabled the possibility of ameliorating the limitations of camera optics and sensors by applying computational photography techniques.

By taking several frames and combining them, one can create low-noise images under low-light conditions or HDR images. A hand-held camera is never perfectly still in the course of acquiring several frames and a frame alignment algorithm is employed to first align the frames before combining them. Adams [Adams et al. 2008] proposed a method for performing real-time viewfinder alignment on smart phones. Our work focuses on extending the original viewfinder alignment algorithm proposed so that it is more tolerant to rotation and object motion.

Although sophisticated image alignment algorithms utilizing feature extractors such as SIFT [Lowe 2004] exists, they are generally computationally expensive. As mentioned in Adams [Adams et al. 2008], a real-time alignment algorithm enables applications such as low-noise viewfinder, HDR viewfinder, panorama capture, and camera-based game controller. Our goal is to extend the viewfinder alignment algorithm without adding significant overhead so that it is feasible to run on modern smart phones in near real-time.

2 Previous Work

There has been significant previous work on image alignment algorithms. One of the best algorithms for detecting matching features across frames is SIFT [Lowe 2004]. A survey of research work in this area is available [Szeliski 2006]. The primary drawback of many of the sophisticated existing algorithms is that they are computationally expensive; thus, they are unsuitable for real-time applications on mobile devices.

Furthermore, highly robust image alignment algorithms are not strictly necessary for our application: we operate under the assumption that images captured in quick succession via the camera's programmable interface are thus relatively similar in orientation and perspective. Implementations of feature-based image alignment are available in OpenCV [Krivtsov], and porting them to a phone is discussed for future work later in this paper, but for now relying on less robust, but significant less compution intensive approaches offer a good performance trade-off.

2.1 Viewfinder Alignment

Adams [Adams et al. 2008] proposed a method for performing real-time viewfinder alignment on a Nokia N95 smart phone. The method works by first computing digests of frames as low-dimensional hough transforms. It uses the digests to get an initial guess of the translation between the frames. Then, it matches corresponding feature points across frames with high Gaussian curvatures base on the translation. Finally, the matching pairs of feature points are used to estimate a rigid motion of the frame (translation with tiny rotation). To match corresponding points, a limit on the distance of corresponding pixel being less than 3 is enforced, which corresponds to a rotation of at most 1 degree.

The aligned frames are averaged and the result is a low-noise image.

3 Methodology

To build a more robust viewfinder alignment algorithm, we explore ways to extend the original algorithm in cases where it is susceptible to failure.

3.1 Object Motion

One problem with the original algorithm is object motion blur. If an object is moving in the scene, the resultant averaged frame would contain several ghost copy of the moving object, which produces an undesirable visual effect to the end user.

Figure 1 shows what would happen if we take the difference between two aligned frames. The text on the computer monitor is changing as we take the picture, therefore we see a large differences between the frames in that region. We also see small degrees of fluctuation over the entire frame due to noise and imperfections in alignment.

If we first blur the image and then take the difference, we get a smoother result as shown in Figure 2.

By thresholding the difference image with a threshold of 20 (corresponding to about 8% variation for an 8-bit channel), a segmentation mask is obtained (Figure 3). We tuned the threshold by experimenting with shots taken under various conditions and observing the distribution of pixel differences.

The mask is expanded to cover boundary pixels between static and moving portions of the frames. From our test, expanding the mask

*e-mail: clfong@stanford.edu
†e-mail:shiweis@cs.stanford.edu

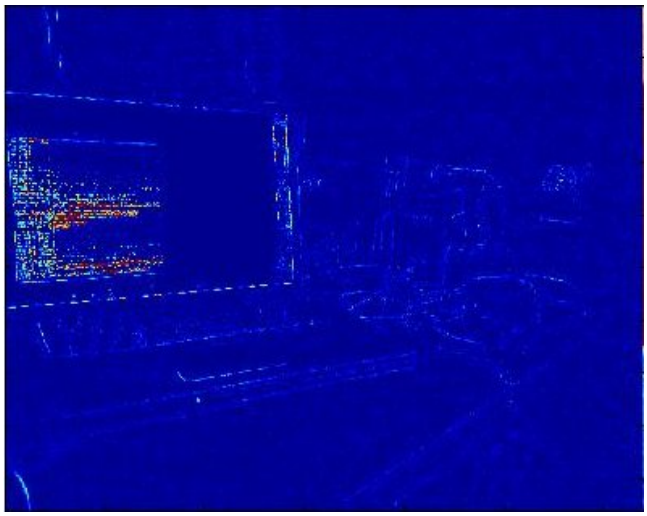

Figure 1: *A difference heat map between two successive frames*

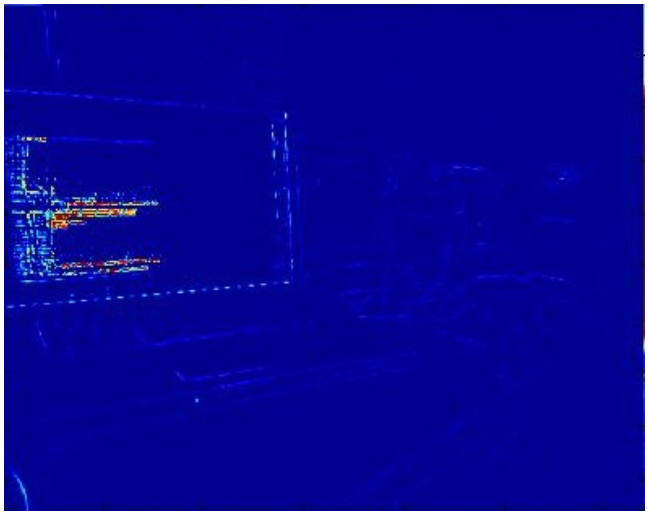

Figure 2: *A difference heat map between two blurred frames*

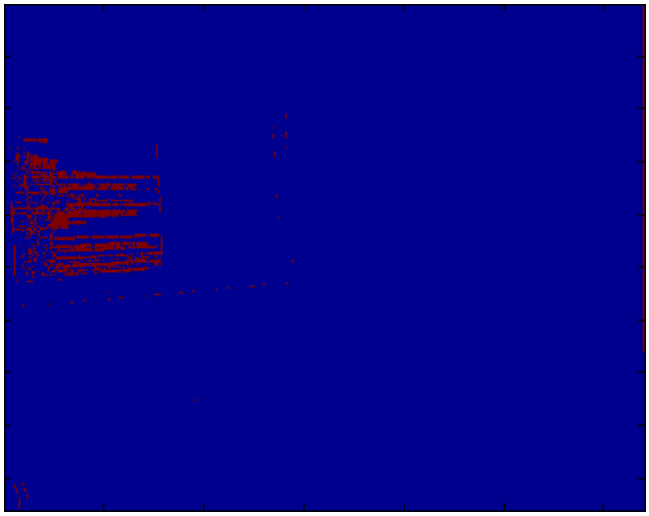

Figure 3: *A mask for segmenting the frame*

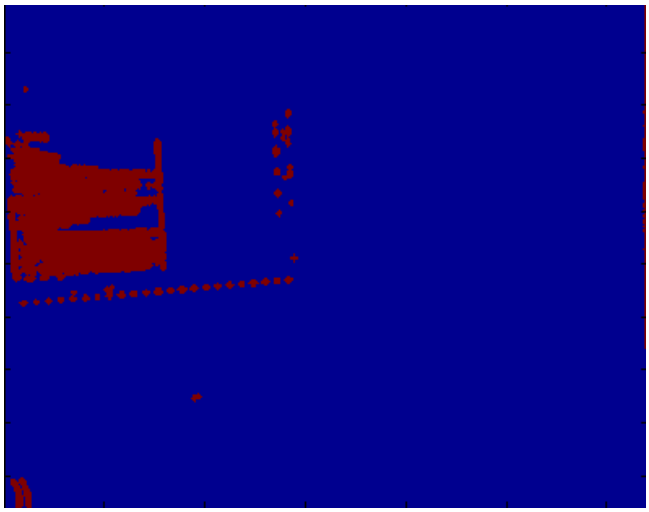

Figure 4: *A mask for segmenting the frame*

by 3 pixel gives good results. Anything beyond that does not significantly improve the final, combined frames. This is shown in Figure 4.

To summarize, we first segment the aligned images into moving and stationary parts. The segmentation is done by blurring the original frames, taking the difference of the blurred frames, and thresholding the difference. Then the stationary parts of the images are averaged, while we only take pixels from a single frame for the moving parts. The final output is a single low-noise image with no motion blur.

3.2 Rotation

Another limitation of the original alignment algorithm is its low tolerance for camera rotation. We propose an algorithm of subregion alignment for handling rotation beyond the one degree limit in the original algorithm.

The rationale behind our extension is as follows: Consider a single pixel in a image that gets rotated, since there is only one pixel, there is no notion of orientation. Any rotation (or in general, rigid motion) of the image can be modeled as a translation at a per pixel level, with a possibly different translation vector for each pixel. Similarly, consider a small subregion in an image that gets rotated, the motion of the subregion can be well approximated by a translation, with minor discrepancies between corresponding pixels (Figure 5).

Our algorithm goes as follows:

- Cut each frame into several subregions
- Align the corresponding subregions using the original alignment algorithm
- For each aligned subregion, find a pair of corresponding feature points
- Use the corresponding feature points from all subregions to estimate the overall rigid motion of the frame

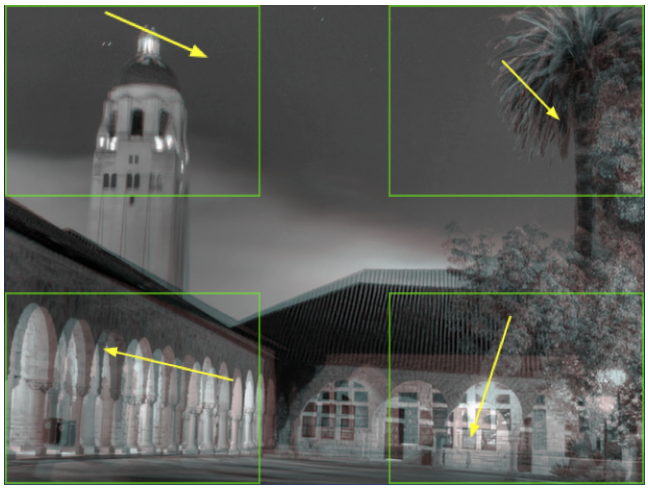

Figure 5: *Rotation as translation in subregions*

3.2.1 Confidence Thresholding

Sometimes the corresponding feature points found can be mismatched for a subregion, which in turn causes inaccuracies in rigid motion estimation (Figure 6). We address this by thresholding subregion confidence so matched feature pixels from low confidence regions are discarded.

Figure 6: *Without confidence thresholding, frames are not perfectly aligned, causing blurring*

For each pair of corresponding subregion, we look at the confidence returned by the original algorithm. i.e. The number of matched feature points returned. If the confidence is above a certain threshold (we used 5 in our experiments), we declare that we have a correct alignment for that particular subregion and pick the best matched point (in 2-norm) from that subregion as the representative matching feature point.

4 Experimental Results

We implemented the ideas stated in the previous sections on the Nokia N900 smartphone to test their feasibility in providing a near real-time improvement of image acquisition on mobile devices. Our code is based on the N800 implementation of the original viewfinder alignment algorithm provided by CS448A (Computational Photography) class.

4.1 Smart Average

While taking outdoor pictures, moving cars often appear as part of the scene, which cause object motion blur to the viewfinder alignment algorithm after averaging. Here we compare the result of combining two consecutive frames using the original algorithm (Figure 7) and using our smart average algorithm (Figure 8).

Figure 7: *A moving car combined using simple average*

Figure 8: *A moving car combined using smart average*

The result shows that we can effectively remove unwanted object motion blur in typical real-world scenarios.

Figure 9 shows that there may still be some image artifacts in regions with soft shadows cast by the moving object or if the object has similar color as the background color. This, however, may be more desirable than motion blur and is not very apparent on small viewfinder screens.

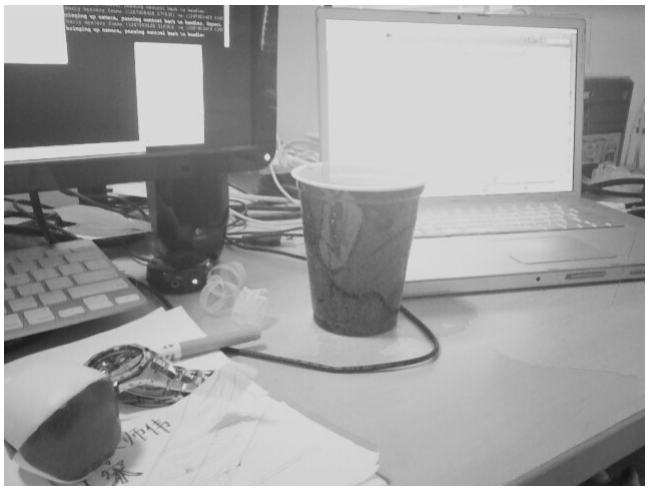

Figure 9: *Artifacts in regions of soft shadows and low object/background color contrast*

4.2 Robust Rotation

The original algorithm usually fails to align images with more than 1 degree of rotation while we found our algorithm to be robust for more than 3 degrees of rotation. Beyond the 3 degrees limit, rotational movement becomes too large to be approximated by translations in the subregions, causing alignment failures. With larger images, smaller subregions can be used to add robustness.

We compare the robust rotation extension to the original algorithm on two frames with a 3 degrees of rotation (with center of rotation in the top-left quadrant). The result shows that our extension (Figure 11) successfully improved robustness towards rotation and was able to handle cases where the original algorithm (Figure 10) fails.

Figure 10: *Combining two rotating frames using the original algorithm*

Due to hardcoded parameters in the original alignment algorithm, the original algorithm implementation also fails for larger sized images. By adjusting the number of subregions and size of each subregion used in robust rotation, our extension allows the use of the original alignment algorithm on larger images without modification.

Figure 11: *Combining two rotating frames using the robust rotation algorithm and confidence thresholding*

4.3 Computational Cost

We also compare the running time of smart average and robust rotation with respect to the original algorithm for aligning and combining two consecutive frames (320 by 240 pixels). The results shown here are the average running time per pair of frames from running 100 tests on each configuration.

We see that while smart average takes more time than the original algorithm (mainly due to blurring of images), robust rotation doesn't not add significant overhead to the processing time per frame. The longer processing time needed for smart average may mean greater object motion between frames, but as long as frames do not have motion blur individually, the combined result should also be blur free.

	Original	Robust Rotation
Original	0.0903s	0.1040s
Smart Average	0.1665s	0.1746s

Table 1: *Running time comparison of Smart Average, Robust Rotation and the original viewfinder alignment algorithm*

We have not spend time optimizing our implementation for the N900, but our result already shows that the extensions can be achieved in near real-time on modern smart phones. As a point of reference, the original implementation runs in 0.0333s on N95 when optimized for that device, compared to 0.0903s on N900 unoptmized.

5 Future Work

Future work may include robust feature-based image alignment that can account for any degree of rotation about the optical axis, as well as perspective changes due to rotation about other axes (pitch and yaw). Such approaches could be readily implemented on a desktop computer with the libraries available in OpenCV, but porting to run real-time on a mobile computing environment would be a challenge [Szeliski 2006].

Another option would be to collect images, but offload them for processing, viable as many phones are wireless network-connected devices. The images could then be processed to any degree without the constraints of the mobile devices memory and processor, and returned to phone for preview and saved remotely to a user's home photo collection.

Future work could also explore more sophisticated algorithms for segmenting motion, such as considering not just aligned pixel differences to classify a pixel as "moving region" or "non-moving" region, but also classification of nearby pixels. Moving object tend to be coherent, so the motion of nearby pixels is an important indicator of motion. Using the approach would reduce the effect of noise and improve motion segmentation for smart combining. Such an approach using energy minimization and graph cut was employed by Rother et al. [Carsten Rother and Blake 2004] to extract foreground image segments, and similar technique may prove fruitful for motion segmentation.

Finally, a variety of smart per pixel combining algorithms could be explored. Pixels from a series of images could be combined intelligently to remove motion blur: pixels inside the moving region need not be from a single image but intelligent combined from different images or a subset of images to reduce noise within these regions of motion.

6 Conclusion

To conclude, a number of successful refinements are developed for more robust viewfinder alignment. The extensions allow the alignment algorithm to handle object motion in the scene, rotation of the camera, and higher resolution viewfinders. These extensions are fast enough to maintain near real-time speed.

Acknowledgements

We would like to express our thanks to Prof. Marc Levoy, with his insightful class of computational photography, as well as providing us with Nokia N900 cell phones to explore our ideas on building better cameras. We would also like to thank Prof. Fredo Durand for his advice on ideas to try to extend the original alignment algorithm. Last but not least, our TA, Jongmin Baek, has also given us very valuable advice in using the FCam library on the N900. We wouldn't have been able to proceed far in our project without his timely email responses.

References

ADAMS, A., GELFAND, N., AND PULLI, K. 2008. Viewfinder alignment. *Comput. Graph. Forum 27*, 2, 597–606.

BENNETT, E. P., AND MCMILLAN, L. 2005. Video enhancement using per-pixel virtual exposures. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, ACM, New York, NY, USA, 845–852.

CARSTEN ROTHER, V. K., AND BLAKE, A. 2004. Grabcut - interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (SIGGRAPH)*.

KRIVTSOV, O. Image alignment algorithms.

LOWE, D. G. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 91–110.

SZELISKI, R. 2006. Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Computer Vision,*.