Expanding *Interactive Digital Photomontage*

Sean P. Dukehart*
University of Maryland Baltimore County
Social Security Administration

Abstract

This paper expands the unique pipelined work flow called **FUSE**, to enable the combination of some of the best features of multiple image synthesis processes into a possible framework. The goal is to develop the flexibility and usability of this work flow into a more fluid process by filling a large gap in the current functionality by the inclusion of an image registration process. Additionally, other novel enhancements are explored to increase the responsiveness of imaging objectives, such as multi-scale graph cutting and alpha transparency region-exclusion.

Keywords: photomontage, registration, panorama, image mosaic, image synthesis,

1 Introduction

In situations where multiple images exist for a given scene, with focus on different parts of the scene from different view points, there are many instances where the need to pull the multiple images together into a unified version of the original scene arises. For instance, a mountain is summited and the climber wishes to capture everything that they see on their camera. It's simple enough to imagine that a large number of pictures would probably result, trying to encompass the entire expanse of the view. Showing the pictures one at a time doesn't do the original view justice. So the creation of a panorama from all of the pictures is a logical step. However, this process generally requires a lot of manual intervention.

Additionally imagine a wedding. The photographer is busy snapping a set photos of the bridal party while the amature photographer, also known as Aunt Fanny, is doing her best to capture the moment. This results in some individuals looking at Aunt Fanny, while others are looking where they were supposed to. The eventual goal would be to combine bits and pieces of these together into the best possible composition resulting in a final image where everyone is looking their best and is facing the desired direction. But the difficulty arises due to the fact that the images were taken by two separate people from two different positions with two different cameras.

Ultimately, a means of not only selecting the best parts, but also aligning and mending a number of separate images into a single image is the desired result. Additionally, to be able to interactively see the results of actions performed on those images provides the necessary feedback to make the system truly useful.

*e-mail: sean.dukehart@gmail.com

2 Related Work

The paper "Interactive Digital Photomontage," [Goldman and Chen 2005] presents a framework that the authors dubbed **FUSE** for the compositing of images through graph-cut optimizations [Boykov et al. 2001] and gradient-domain fusion [Fattal et al. 2002; Parez et al. 2003]. In essence, their process enables a user to stitch together a number of images based on imaging objectives to produce a final composite image. Seams which would be the least obvious transition from one image to the next are derived and used to cut desired regions of individual images out. While this process is incredible, it lacks the ability to do any sort of practical image registration, thus requiring that all images be from basically the same viewpoint, or have registration done externally from the application. Additionally, the process tends to be slow, and is extremely heavy weight for large images.

Since graph-cut optimization has been recognized for its producing of such high quality results, it has found large adoption in a wide range of applications. Due to its natural complexity, large max-flow/min-cut computations are quite taxing. Thus, there was a need for optimizing it to get quality results with a more rapid turnaround. In the tradition of using multilevel approaches to solving problems in the imaging realm, Lombaert et al. [2005] came up with a method for applying the same principles to graph cutting. They described a methodology for the calculation of graph-cuts on low resolution versions of a high resolution source image, and then upsampling and correcting the nodes that fall along the cut. We adapt this method to fill a much needed performance gap in the digital photomontage framework.

This framework has also been used in conjunction with additional techniques to remove "vignetting" and exposure variations across multiple images [Goldman and Chen 2005]. The focus in this paper was more on the color variation between and across image edges, though it provides useful insight into extensions that would be of benefit to include in this application. Its use of an external registration application named **AutoStitch** led to the insight that it might be used for here for similar purposes.

AutoStitch was the creation of David Lowe [1999; 2004], in which he implemented a version of his SIFT algorithm for multi-image registration with an allowance for RANSAC filtration. This application provides a fully automatic solution to feature detection, image registration, and panorama blending. However, the necessary interactions for maintaining of separate, yet registered images was found in another implementation of the SIFT algorithm with available source and much more immediate flexibility, namely Sebastian Nowozin's **autopano-sift**. Alternatively, there are newer, and supposedly more robust feature detection algorithms such as SURF [Bay et al. 2006], though the SIFT algorithm is the most widely used and was used to implement the automatic feature detection portion of the image registration process in this paper.

There are a number of techniques for image registration to choose from [Brown 1992; Szeliski and Shum 1997]. Many provide implementations that also include blending methods for the merging

of the registered images. While some are good at the registration, there seems to be a lacking in the actual compositing arena. A rather full-featured open-source registration implementation produced by Helmut Dersch, **Panorama Tools** has the potential for such good results that a number of different graphical interfaces have been developed for it. The implementation that stands out due to the feasibility to incorporate different feature detection algorithms right into it is **hugin**, which is actually a project being supported by Google's Summer of Code project. The benefit of this application is that it enables the use of **autopano-sift** for the feature point detections, and also has ties into additional lens corrections and vignetting compensation. All details pertaining to the image registration process are directly correlated to **hugin** functionality.

While other papers have focused on strictly creating panoramas from video [Agarwala et al. 2005] or still images [Szeliski and Shum 1997], or on taking images and registering them into a scene to provide interactive walkthroughs of the scene [Snavely et al. 2006] (provides the foundations of the current Microsoft Live Labs *Photosynth*), we aim at bridging the gap between related images and the minds remembrance or conveyance of a better than reality moment in time for the scene those images portray. Through the incorporation of the **SIFT** algorithm with the image registration of **Panorama Tools**, the digital photomontage pipeline gains that which it was lacking.

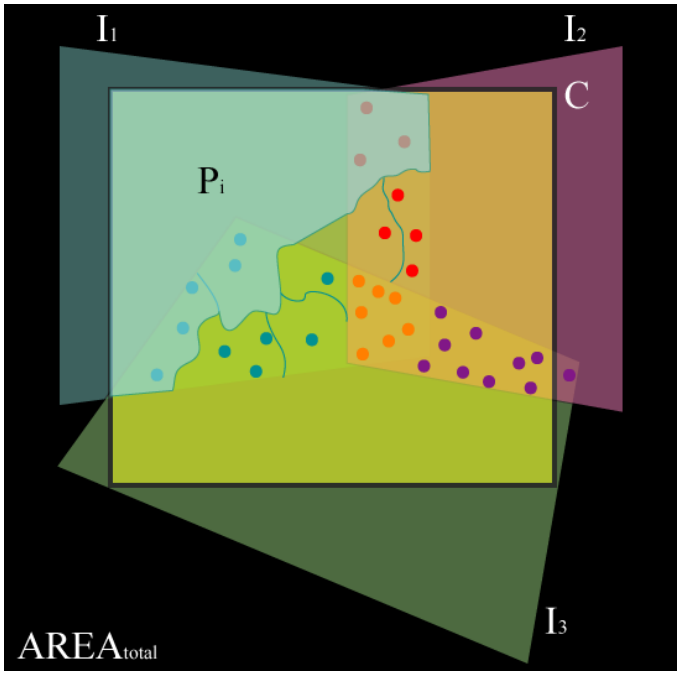

Figure 1: *This figure is an overview of the process that this paper presents.*

Base Image Size	% Transparent
970 × 1400	30%
1200 × 830	45%
1200 × 830	46%
840 × 1240	47%
1000 × 675	61%

Table 1: *Base image size to aligned image size comparisons showing how much transparent padding is necessary for a single image to be aligned (aligned images size with padding is 1200 × 1800)*

3 Implementation

3.1 Overview

The following gives a brief overview of the new incorporation into the process detailed in this paper.

1. Image Registration
   (a) Load initial images I_1 - I_n
   (b) Generate control points using computer vision-based feature detection algorithm such as SIFT
   (c) Refine control points until an acceptable level of error is reached
   (d) Align images based on corresponding control points
   (e) Optimize image relationships accounting for lens distortions
   (f) Generate warped / aligned images or image transformations
2. Image Objective Processing
   (a) Load aligned images or transform initial images and determine NULL / Alpha regions (if possible)
   (b) Rescale transformed images, padding if necessary, to fit into image pyramid (mipmapping concept)
   (c) Perform graph cutting based on imaging objectives taking into consideration NULL regions, based upon current scaling factor

3.2 Image Registration

As can be seen by the example presented in Figure 1, the Image Registration process starts with initial images I_1 - I_n, and proceeds to generate control points that reflect relationships between overlapping regions of the images. For example, the purple and orange dots reflect the relationships between I_2 and I_3. Once these points have been refined and optimized based on properties of the lens on the camera that took them (generally for digital images, this information can be gleaned from the EXIF attributes), image transformations / warping can take place to actually align those control points.

Due to the fact that the transformed images don't necessarily all fall within the same space (when the images are not time-lapsed tripod-based), in order for the images to actually align, the entire canvas that the transformed images reside in needs to fit all of those images. Thus AREA_{total} can be significantly bigger than any of the single images that it encompasses. But each of those images must fall within that area for their relationship to other images be known. Thus the simplest solution is to fill non-image regions of AREA_{total} for I_n with a NULL color, black in this case. In effect, *n* images of size AREA_{total} result.

Since **hugin** provided a very clean interface for working with and manipulating images to move them toward alignment, it was kept as an entirely separate piece of the pipeline, with minor modifications being made to allow for its generation images that could be used in the later process presented in Section 3.3. The large portion of the work that was done was to optimize **FUSE** for its acceptance of the output that this stage produces.

3.3 Image Objective Processing

As the majority of this project is built upon the preexisting framework presented by Agarwala et al. [2004], namely the **FUSE** application, when considering the inclusion of aligned images it immediately became clear that a few changes to improve performance would be necessary. Since the simple image transformation case where an image's relationship with other images is dictated by its location within AREA_{total} results in *n* images of size AREA_{total}, the processing of images when AREA_{total} is large became difficult. As the composite in **FUSE** encompasses AREA_{total} and results in the operating on *n* images of size AREA_{total}, storage limitations as well as the magnitude of processing time when considering graph cut computations across multiple images to accomplish the imaging objectives becomes a large bottleneck.

As shown in Table 1, when an alignment of images completes, a large portion of the image data is just wasted space (upwards of 60% or more in some cases). Since this space isn't providing any useful purpose except to allow the images to all align, there is no real reason to treat it as valid data. Therefore, viable solutions to the NULL space problem were considered.

3.3.1 Cropped Image Computation Restriction

In pondering possible solutions, the simplest optimization that presented itself was to crop the transformed images to *C*, or the maximum region that encompasses no NULL space. This could be done during the image registration stage as a post process. While this may greatly reduce the overall size of all the images, it could potentially not be desirable in some instances where a specific feature may reside outside of *C*. Therefore, forcing an automation of the cropping process is most likely not the ideal solution. Instead, windowed crop regions were added.

Windowed crop regions provide arbitrary unrestricted regions for specifying what a desired result might consist of. The major benefit of this was in the fact that computations across the images are restricted to these regions. Thus, when only a small portion of the overall canvas is desired, very significant speed-ups will be seen by enclosing the desired work area in a crop region. Note that the cropping operation is 100% non-destructive meaning that it only affects the current computation, and can be resized to get different desired results. Any reduction in AREA_{total}'s size toward *C* or smaller will drastically cut down on the amount of image data that must be considered for the energy cost cutting.

This was implemented as a restriction put on the energy computation and minimization process. Since these computation are generally calculated over an entire image to minimize the cut for that image, by specifying a sub "crop" region of that image as the extents to which computation should take place, calculating the image's energy, and from there the best cuts to reduce it was simply a matter of ensuring that both elements operate on the same space. Thus, by having the energy functions be tied to the cropped region, we restrict which pixels / nodes will be added to the graph for graph cutting.

Although this results in significant speed ups for calculations over an image when the cropped region is set toward *C*, it should be noted that not all alignments will result in large regions of NULL space, and it will thus not be desirable to perform cropping as the whole composite's surface area may be filled with useful image data. It therefor became apparent that to further increase the processing speed of calculating graph cuts, additional enhancements would have to be considered.

3.3.2 Alpha Stoke Constraint

Due to the nature of **FUSE**, and its being a stroke based application in which the user paints "strokes" on a series of images to designate desired regions or features, stroking restrictions also were quickly found to produce significant computation reductions. In the case where the entire surface of the composite is covered with useful image data, a stoke placed anywhere on that composite will provide the necessary information to allow a cut to be computed. But when the entire composite is not comprised of pertinent information, namely there are regions of NULL or only Alpha information that won't productively contribute to the final image, we can use this fact to restrict strokes.

By enforcing the rule that all strokes placed on the images will only be considered if they lie within image regions, we can prevent situations such as when a user places a stroke on a NULL region entirely. In such an instance, the stroke is merely discarded and no further processing is necessary.

In the complimentary case, a stroke lies partially on image data, while it lies partially on a NULL region. This, should the entire stroke be taken, present a challenge. In terms of graph cutting, cuts through a NULL region all presumably have the same cost as any cut made will not be seen. Thus, trying to terminate the max-flow becomes a challenge, and produce seedy results. The simple and robust alternative is to truncate all stroke information that extends outside of the imaged regions. Thus the graph cut algorithm has a logical boundary along the edges of where the image data stops, and the NULL region starts.

3.3.3 Multilevel Banded Graph Cuts

The most significant enhancement to **FUSE** was the incorporation of the "Multilevel Banded Graph Cuts Method" from Lombaert et al [2005]. This method comprises a 3 part process for fast image segmentation, namely "coarsening," "initial segmentation," and "uncoarsening."

1. Starting with image I, downsampled images $\{I_0, I_1, ...I_K\}$ are generated through a coarsening process
2. A graph cut is performed at level K on I_K to generate a binary boundary image
3. The binary boundary image is used at level $K-1$ to seed the locations for the "banded" graph cut
4. A banded graph cut is performed for each subsequent level using and improving the results of the prior level to reach an "uncoarsened" result

The authors found that the banding was absolutely necessary to see any speed improvement in using this approach as opposed to regular graph cutting. Although the full algorithm wasn't implemented (namely the banding portion), by merely performing the graph cutting at a downsampled resolution, significant speed up was seen without severely harming the integrity of the final images.

Due to the possibility of arbitrarily sized images, power of 2 coarsening is achieved by first resizing the images to have size divisible by the desired coarsening level. Rather than scaling here, the image is merely padded with extra rows/columns to ensure that the initial image data remains unscathed. To scaling is done by means of box averaging. Once the intermediate scaled images are generated, the amount of quality/speed is selected. It should be noted that the authors observed noticeable artifacts when choosing an amount of scaling greater than 3 levels. Thus, its a trade-off between speed and quality, where level 0 results in no speedup but performs its calculations on the initial images, and level K will produce some

Figure 2: *Cropped resultant image after registration and selective fusing*

Figure 3: *Patches of 4 different images used to generate Figure 2 (the red regions indicate where NULL space exists for this particular composite)*

order of magnitude of speedup, yet will produce visual artifacts. It was observed that levels 1-2 produce fairly decent results without the banded "uncoarsening," while the banded uncoarsening would surely improve the results (though how much it would increase the compute time is unclear).

4 Results

The results are quite compelling and show that there are many directions to seek improvement for this particular style of problem, namely finding what needs to be computed and what doesn't. The extension of the registered images concept within **FUSE** has opened up more pressing areas of study, such as additional graph cut optimizations for this type of images. This led to the incorporation of a Multilevel Graph Cut algorithm which in turn produced significant reductions in compute time as well as memory consumption. Additionally, by manually specifying what area of a composite is desirable (by cropping), additional speed ups can be seen due to restricted computation.

5 Future Directions

Possibilities for future work include an alternative or possibly complimentary approach to the calculation process in the form of tiled graph cuts. However, the feasibility of this with the whole concept of graph cutting is yet unclear. Additionally, in dealing with composite regions with a large AREA_{total}, capturing fine scale detail would be of the utmost concern. However, working in multiple scales (zoom-levels) is also desirable. In such situations, it may be possible to aggregate the cost of doing both by leveraging the multilevel images to enable the calculation of graph cuts at intermediate scales (i.e. mip-mapping). Finally, the implementation of the uncoarsening through banded graph cutting should be a straightforward enough addition, and would be interesting view in quality / speed contrast to just using the Multilevel graph cuts.

References

AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUCKER, S., COLBURN, A., CURLESS, B., SALESIN, D., AND COHEN, M. 2004. Interactive digital photomontage. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM Press, New York, NY, USA, 294–302.

AGARWALA, A., ZHENG, K. C., PAL, C., AGRAWALA, M., COHEN, M., CURLESS, B., SALESIN, D., AND SZELISKI, R. 2005. Panoramic video textures. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, ACM Press, New York, NY, USA, 821–827.

BAY, H., TUYTELAARS, T., AND GOOL, L. J. V. 2006. Surf: Speeded up robust features. In *ECCV (1)*, 404–417.

BOYKOV, Y., VEKSLER, O., AND ZABIH, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell. 23*, 11, 1222–1239.

BROWN, M., AND LOWE, D. G. 2003. Recognising panoramas. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, IEEE Computer Society, Washington, DC, USA, 1218.

BROWN, L. G. 1992. A survey of image registration techniques. *ACM Comput. Surv. 24*, 4, 325–376.

FATTAL, R., LISCHINSKI, D., AND WERMAN, M. 2002. Gradient domain high dynamic range compression. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 249–256.

GOLDMAN, D. B., AND CHEN, J.-H. 2005. Vignette and exposure calibration and compensation. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, IEEE Computer Society, Washington, DC, USA, 899–906.

HOIEM, D., EFROS, A. A., AND HEBERT, M. 2005. Automatic photo pop-up. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, ACM Press, New York, NY, USA, 577–584.

JIA, J., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2006. Drag-and-drop pasting. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, ACM Press, New York, NY, USA, 631–637.

LOMBAERT, H., SUN, Y., GRADY, L., AND XU, C. 2005. A multilevel banded graph cuts method for fast image segmentation. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, IEEE Computer Society, Washington, DC, USA, 259–265.

LOWE, D. G. 1999. Object recognition from local scale-invariant features. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, IEEE Computer Society, Washington, DC, USA, 1150.

LOWE, D. G. 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60*, 2, 91–110.

PAREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, ACM Press, New York, NY, USA, 313–318.

SNAVELY, N., SEITZ, S. M., AND SZELISKI, R. 2006. Photo tourism: exploring photo collections in 3d. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, ACM Press, New York, NY, USA, 835–846.

STRENGERT, M., KRAUS, M., AND ERTL, T. 2006. Pyramid Methods in GPU-Based Image Processing. In *Workshop on Vision, Modelling, and Visualization VMV '06*, 169–176.

SZELISKI, R., AND SHUM, H.-Y. 1997. Creating full view panoramic image mosaics and environment maps. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 251–258.

Figure 4: *The unblended version of Figure 6, in which the Multi-level "Coarseness" attribute was set to 2 levels of downsampling so as to produce quicker results*

Figure 5: *Patches of 4 different images used to generate Figure 6*

Figure 6: *A final Poisson-blended composite image*