

**REAL-TIME IMAGE REGISTRATION
AND NOISE REDUCTION**

A PART OF THE PEN PROJECT

**SPRING SEMESTER 2003
INDEPENDENT RESEARCH**

**BY
WILKIE KIEFER**

**ADVISOR
PROFESSOR SZYMON RUSINKIEWICZ**

ABSTRACT

This paper presents a method to register a series of images captured by a high resolution camera in real-time. The proposed algorithm attempts to balance accuracy and efficiency properly in order to meet both the time constraints and precision requirements of the Pen Project. The results obtained are promising and possible extensions to this research are discussed.

INTRODUCTION

The registration of multiple images using only the information present within each image is a difficult and challenging task. There are many factors that make a perfect alignment hard if not impossible to obtain: camera noise, perspective distortions, lighting differences, and movement, to name a few. Fortunately, image registration is a well-researched problem and there are numerous approaches and methods for solution. Most of the existing methods attempt to improve registration by creating a controlled environment during image capture, removing possible sources of confusion. For instance, using a very bright and diffuse room while capturing images would remove brightness inconsistency across all the images. The existing registration methods are also all context-specific. Registering a series of scenery images taken by a still camera into a wide panorama requires a different approach than registering a series of images of a single object taken from varying camera locations. This research, following the fate of previous work, will be both context-specific and attempt to improve registration by creating a controlled environment. However, the context of this research is quite different from any other registration problem and therefore provides ample room for innovation.

Specifically, the research proposed below attempts to find a solution to the image registration problem in a real-time environment. With accuracy as paramount, the majority of image registration methods require a lot of time for computation. For most applications, this is fine. There are no time constraints so computation can be done off-line and take as long as needed. However, for this project time will be a consideration and a determining factor during algorithm development. The true challenge of this research will be to adequately balance accuracy with speed. A fast image registration method has little use when the results are inaccurate.

An efficient and accurate method to register multiple images in real-time, however, does

have many interesting applications, especially a field such as computer vision. Acquiring a 3D model in real-time using a camera, for example, could easily take advantage of a fast image registration method. Real-time registration would also be useful in image-based rendering. One could imagine an image-based walkthrough environment that would take advantage of fast registration techniques.

This paper will first describe the context of this research and present the general method developed. Next, after a detailed look at the methodology, results will be presented and discussed. The goal of the paper is to not only explain the various decisions made during development and implementation, but to examine the strengths, weaknesses, and possible extensions of the method using experimental results.

PREVIOUS WORK

Although there has been abundant research on image registration in general, very little work has been done regarding image registration in a real-time environment. The wealth of knowledge on this diverse subject, however, provides a great starting point to develop an appropriate algorithm. In fact, the work of Kuglin and Hines[7] served as an initial test for the proposed method below.

PROJECT OVERVIEW

Currently, the only way for an artist to enter hand drawn art into a graphics program is to use a writing tablet or similar product. These tablets are difficult to use and rarely capture the intricacies of hand drawn images. The main problem that makes tablet input methods difficult to use is the fact that the visual feedback from hand motions is not near the pen, where it should be intuitively, but is located relatively far away on the computer screen. Drawing in this environment is not as accurate or as easy as simply drawing on a piece of paper. Freeing the artist from these difficulties is one main motivation behind what is currently referred to as the Pen Project¹. The goal of the Pen Project is to allow an artist to draw directly onto a piece of paper and have the resulting images displayed immediately on the screen. This will not only free the artist from the constraints of a tablet but allow the computer to capture the variations and intricacies of charcoal or pencil on paper.

¹ A postdoctoral project initiated by David Bourguignon

The project has been divided into parts as shown below (Fig. 1). The main idea is to have two or more cameras tracking the position of the pen or pencil on the paper. The process responsible for the tracking will then send the pen position to a higher resolution camera that will follow the path of the pen. The position coordinates will also be broadcast to other processes. The output of the high resolution camera will be fed directly to my project, the image registration and noise reduction process. Here images come in at around 30 frames per second. Each group of five images is cleaned up, registered, and combined to form one noise free high quality image. This image is then fed to the next process which aligns the images from my process globally. Finally, the full image is exported to the application layer for the user to see and edit.

Examining the basic functionality of the Pen Project brings to light the two main challenges of my project: accuracy and efficiency. On one hand, if the images output from my process are not accurate, the aforementioned goal of the Pen Project to capture the intricacies of hand-drawn art is lost. On the other hand, if it takes too long to process the images, any aspect of real-time control is lost to the user. If the Pen Project cannot function relatively quickly, there will be long delays between the time the user makes a mark and when that mark shows up on the screen. Working in such a delayed environment is frustrating and difficult.

The details of the Pen Project need not be continued here, however, the time constraints given to my algorithm are defined by overall project. As a rough time constraint calculation, receiving images at 30 frames per second and exporting images at 6 frames per second allows my algorithm 0.167 seconds to register and blend each group of five images. The impact of this time constraint will be examined in detail later.

As a note, the research described up until now has focused on the problem of image registration. However, as seen from Figure 1 below, noise reduction is an important aspect of the algorithm as well. Because it provides a more difficult problem, registration will be the topic of the majority of this paper. However, the implementation and importance of noise reduction will be addressed in the forthcoming methodology and discussion sections.

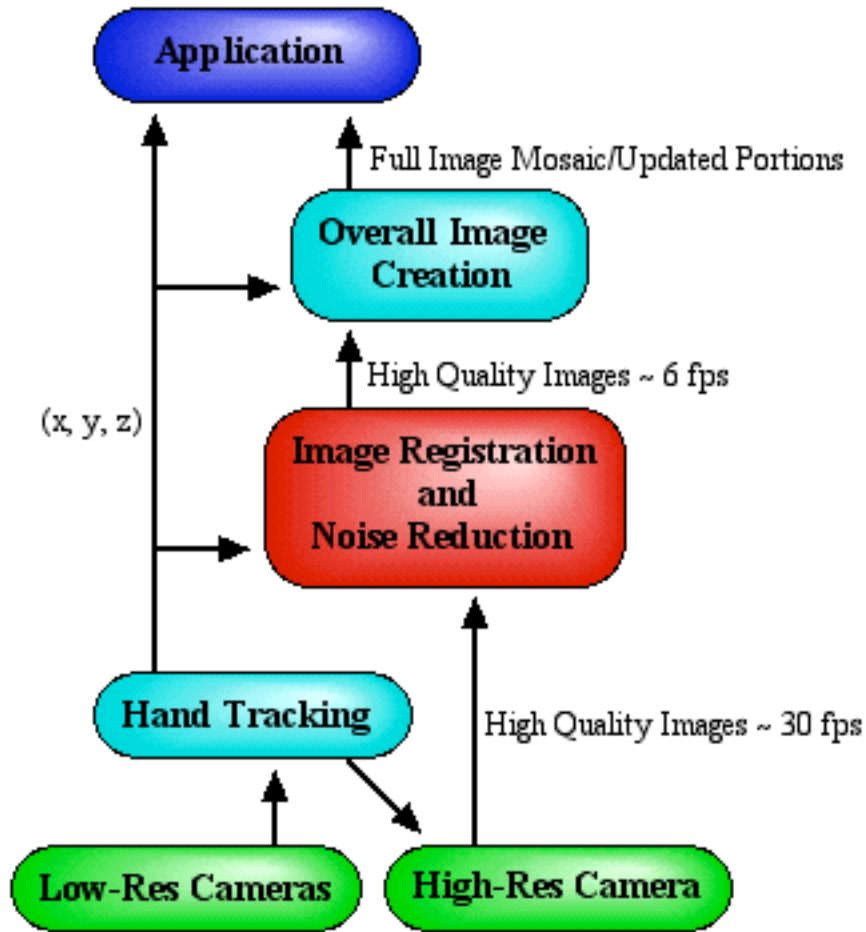


Figure 1

APPROACH

The first approach I used to align images was a common registration method entitled Phase Correlation[1][7]. which takes advantages of the properties of the Fourier Transform. In brief, the Fourier Transform is a complex function that takes each point from the image domain and translates it into the frequency domain. The basic idea behind phase correlation is that displaced similar images have the same frequencies but are out of phase with respect to each other. Using peaks in the frequency domain, one can calculate the displacement between two images and translate this displacement back into the image domain. Then using the newly calculated image displacement, one can align the images. This property of the Fourier Transform works for rotations as well as translations, but in the case of rotations requires more calculation.

After implementing this approach, I obtained very accurate displacements to sub-pixel values. To calculate the Fourier Transform and inverse Fourier Transform I used FFTW[8], an award winning C subroutine library developed at MIT and known for its speed and performance. However, even using FFTW, the calculation times were too slow. On average, it took about 0.9^2 seconds to get the displacement between two single images. To align five images using this method would take around 3.6 seconds (only four displacements required to align five images), not anywhere close to the required 0.167 seconds. Although the results were extremely accurate, I had to use another approach because the time required to calculate the Fourier Transformations were too great.

The approach used in my final implementation is comprised of two main steps. The first step is registration and uses a pyramid comparison scheme. Each input image is scaled to fifty percent its original size n times, where n is a user defined depth. This pyramid of images is stored in an image list. To compute the displacement of one image with another, image lists are compared. To compare two image lists, the two smallest images first are compared using sum of pixel differences squared over a k by k displacement matrix. Again, k is user defined parameter. In other words, if k was 3, the two images would be compared over a 3 by 3 area where one image remains static and the other is displaced by one pixel up, down, left, right, each of the four diagonals, and in the center position. At each position, the average difference squared per pixel is calculated. Using these numbers, the actual displacement between images is calculated. This displacement is then propagated up the image list and the next two larger images are compared utilizing this previous displacement. Each pixel an image is moved at level n correlates to two pixels moved at level $n - 1$. Using this pyramid scheme, an image can be moved large distances with relatively few comparisons initially. As n decreases, the moving image approaches a match and more comparisons are calculated to determine smaller movements. Finally, using paraboloid minimization on level 1, sub-pixel displacements can be calculated. The image below is an example of two images displaced by 14.3 pixels. The displacements and total image movement are shown at the top.

² As a note, all times listed in this paper are taken from tests on the hats.princeton.edu server.

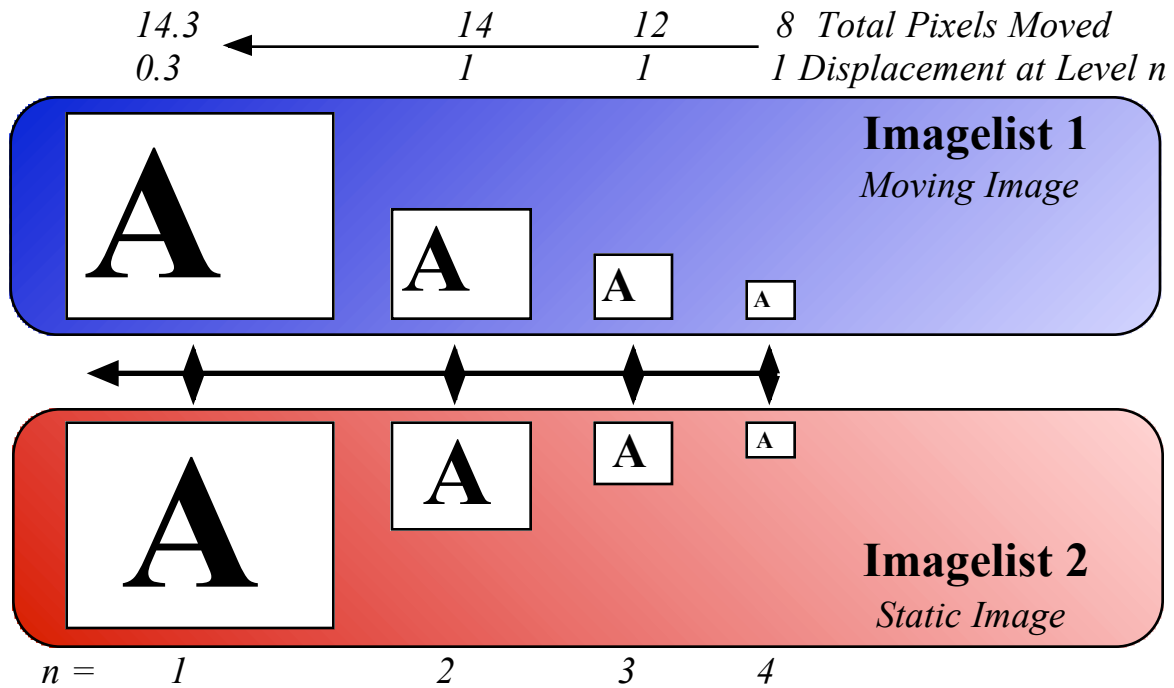


Figure 2

The second stage of my approach simply consists of blending the five images together according to their relative displacements. To do this, I use bilinear interpolation to transform each image into the same coordinate system. Once this is done, blending is achieved by simply averaging pixel values over five images for each pixel. This second stage of the process can easily be adapted to include enhanced noise reduction, image sharpening, or resolution increases.

METHODOLOGY

The first major issue encountered during implementation actually had to do with the input images. The high resolution camera used for the Pen Project produces interlaced images. This means that when an image in the camera is refreshed, every other row of pixels is replaced. Then the other interlacing rows are replaced. When the camera is moving fast, this discrepancy between pixel rows is obvious and does not look good. In addition, it obscures the detail of the hand drawn lines. From here on, I will refer to the artifacts produced by this phenomenon as scanlines.

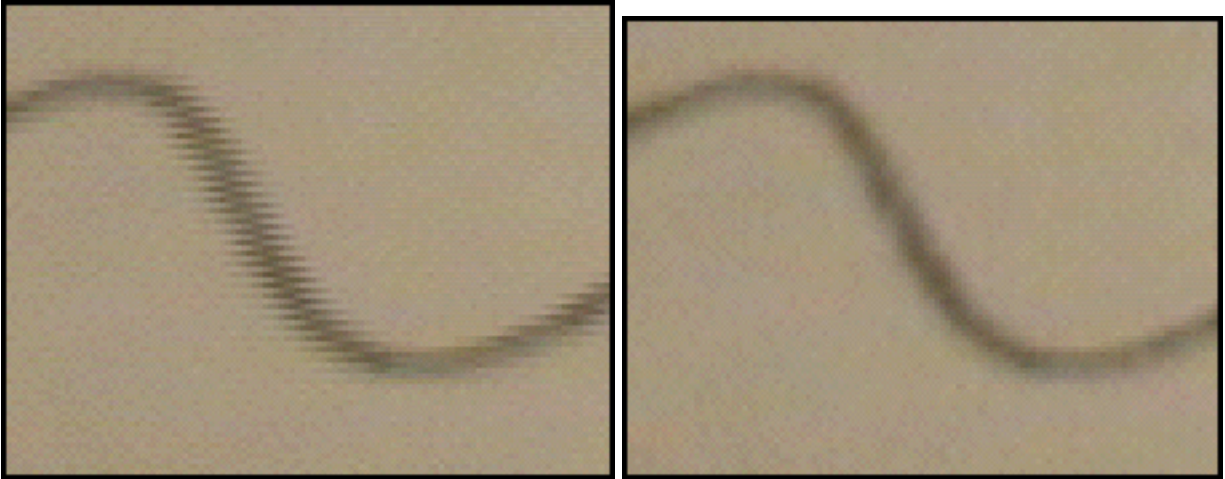


Image 1. Image directly off camera (2x)

Image 2. Image after scanline interpolation (2x)

Initially, in order to combat the scanline situation, I only considered every other row when comparing images with each other. However, I abandoned this strategy for multiple reasons. First of all, calculating vertical displacement between two images using every other row was not very accurate. A displacement of 1 pixel vertically was difficult to determine, even with sub-pixel calculations because there was not enough data. Second, I realized that eventually I would have to confront the scanline problem when blending the final image. Avoiding it during comparison would not make the issue go away. Thus, in the final implementation, I dealt with scanlines before anything else and did all future comparisons over every row and every pixel.

To actually fix the scanline problem, I removed every other pixel row and used linear interpolation to fill in the newly empty rows. My implementation supports two types of interpolation, simple and complex. In simple interpolation, the missing pixel is an average of the pixels directly above and below it. In complex interpolation, the missing pixel is calculated using the surrounding diagonal pixels as well as the pixels directly above and below it. This allows for a smoother image overall. An example of complex interpolation is shown in Image 2 above.

Once the scanlines have been removed using linear interpolation, I create the image lists mentioned above. In this implementation, the user can specify the depth of each image list, n , as well as the size of the displacement matrix, k . In order to create the image lists, each image must be scaled down by fifty percent n times. To scale down each image, I simply take groups of four adjacent pixels that form a square and average them to produce one resulting pixel. Doing this for every group of four pixels results in scaling the image by fifty percent. This is a fast and simple

scheme to scale an image down, but could easily be replaced with any another method. Gaussian sampling, for example, could be used instead for more accurate results. However, this sort of scheme is computationally more expensive.

Next, the five images are labeled 1 through 5 with respect to the order of arrival. Image 3 is branded as the ‘center’ image to which all other images are aligned. To align all 5 images requires four comparisons. Image 1 is compared to image 2, using their respective image lists, and image 2 is compared with image 3. Images 4 and 5 are compared in the same way to image 3. With these four comparisons, there is enough information to align all five images.

Before going on, it is important to examine the details of the image list comparisons. The basic functionality was described earlier, however the details of sub-pixel calculations were left out. On all levels of the image list above $n = 1$, the average differences squared per pixel are calculated over the k by k displacement matrix. To choose which direction an image should be moved for maximal alignment, the direction of minimum differences squared is chosen. This is good for levels above $n = 1$ where just a general direction movement is needed. However, for sub-pixel calculation on level $n = 1$, the minimum value does not suffice. Because pixels are just samples of a given area, the images off of the high resolution camera are offset at values that have nothing to do with pixel integers. The offset is determined by the position of the camera in space. Thus, to truly align images accurately, sub-pixel calculation is needed.

To calculate sub-pixel offsets, my implementation fits a paraboloid to the data values in the displacement matrix using Magic Software's open source least squares fitting algorithm [2][3]. The x and y values are equal to the integer displacements in the matrix, and the z values are equal to the average pixel differences squared at each x and y location. An interesting tradeoff appears when examining this process. As k increases, there is more displacement data, the paraboloid is more accurate, and thus the sub-pixel calculation is more accurate. However, an increased k value comes at a higher computation price. On the other hand, to some extent, the difference between a pixel offset of 0.01 and 0.02 is arguably not noticeable, so extremely accurate calculations might not be necessary. A difference of 0.0 and 0.5 is noticeable, however, so sub-pixel alignment is still an important step.

Once the paraboloid is determined, I use the GNU Scientific Library[5][6] minimization functions to iterate over the paraboloid until the gradient is zero. For my implementation, if the minimum of the paraboloid is not found after 100 steps, which usually means the paraboloid is

not a good fit, the alignment defaults to the x and y integer values with the minimum average pixel difference. In addition, if the minimum is found outside of the k by k range of displacements, I discard the value and default to the minimum x and y integer values mentioned before. I feel that the paraboloid is not accurate in areas with no data and should not be used for sub-pixel alignment outside the data range. In addition, when the minimum is outside the displacement matrix range, it suggests that the images do not align anywhere over the displacement matrix and further movement is required. In such a case, one should increase n or k to allow for greater movement. Another possibility, which my implementation did not do, would be to automatically move the image in the proper direction until the minimum is found within the matrix range. This would assure that the images were moved far enough even if the user specified a n and k that were too small.

After the five images, mentioned earlier, have calculated their displacements with regard to the center image, image 3, they are ready to be aligned. Images 1, 2, 4, and 5 are all transformed into the image space of image 3 using bilinear interpolation sampling methods. Bilinear interpolation sampling was chosen as a middle ground between point sampling, which produces poor results, and gaussian sampling, which is computationally expensive. Points where the images don't overlap with image 3 are replaced with black pixels. Once the images are all in the coordinate system of image 3, blending the five together is very simple. For every pixel in the output image, pixels from each of the five original images are averaged together. Because the original images have been translated with respect to image 3, the area of overlap will not be 100 percent. Thus, the output image can be trimmed in areas where all five images did not overlap.

At this point, the output image is complete. Basic noise reduction is taken care of by the fact that five images are blended together. Camera noise can be reduced through this simple blending. However, additional noise reduction techniques can be applied at this blending stage.

The main assumption made throughout this method is that the input images are misaligned by translation only. This assumption was made for mainly two reasons. First, the images are coming off of the camera at high framerates and are therefore close to each other in the physical world. The warps due to perspective changes should be minimal. Secondly, we assume that the camera is placed in such a way that it minimizes all movements except for translations. Thus, for this model, the pyramid scheme should perform well. In addition, since the method was still under development, it was much faster to get things up and running to find results. This

developed pyramid scheme would not handle rotations as it stands now, but could be adapted to do so. One suggestion that was given to me was to calculate rotational displacements in a similar manner to the current translational displacements. This could be done once the image was translated to the best approximate place. However, for the purposes of this research, we assume only translated images.

RESULTS

When testing my implementation, I was basically looking to evaluate how my method handled the two big challenges it faced: accuracy and efficiency. However, these are both broad categories and need some further definition. As far as efficiency, I wanted times that were in the right ballpark, so to speak. The constraint was determined by the Pen Project framerates and set to 0.167 seconds to compare five images and output one. I aimed to be able to do that in under 0.5 seconds. This amount of leeway would allow me to develop a fast algorithm and not worry about too many of the intricate details of optimizing my code. For accuracy, first and foremost I wanted the alignment method to work. I wanted minimal blurring and accurate displacement measurements. Secondly, I aimed for consistency. If only every third comparison worked accurately, the method would not be very practical.

With regard to time, my initial results looked very promising. I was able to create and compare two full size camera images of 720 by 480 pixels at a image list depth of 3 or 4 in roughly 50 milliseconds. When I was initially only comparing every other row, my times were even faster. However, as my implementation neared completion the times began to slow down because I was adding much more computation. For example, I added the scanline interpolation and the image translation transform. With my final implementation, comparing five images and blending the adjusted images into one final image takes over 1 second. With larger values of k and n , the full process can take ten seconds or more. However, with parameters set to normal values, the full process takes just over a second.

I can attribute the slow down to two things mainly: the additional functionality and computation added toward the end of implementation mentioned before and the fact that I spent less time on code optimization and more time on accuracy toward the end of my implementation phase. I also wasn't able to test my code on a fast dedicated machine. Although the times are

slower than my goal times, I feel that this method can be made to attain the proper speeds. There are many options to speed things up, however, their discussion will be saved for the next section.

My next tests were for accuracy. Basically, I took real data straight from the camera in the style of the pen project. I had two series to work with, a fast panning series with high scanline levels and a more static camera shot with use of the zoom. The main traits I looked for were good alignment and consistency across input samples.

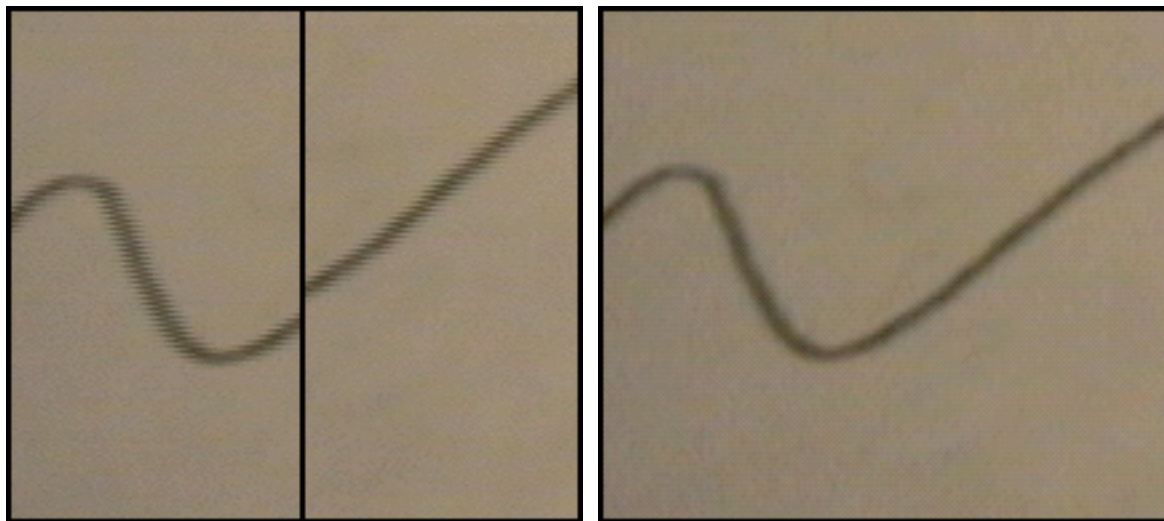


Image 3. Examples from input series of 5 images from the camera. Image 4. A section of the resulting image

Image 3 contains two example sections from a series of 5 images received from the camera while it was panning quickly across a page. Image 4 is the result from my algorithm³. Notice that the scanlines are gone and the image is well aligned. There is minimal blurring along the drawn line. The lack of line definition seen is actually due to a very dull marker rather than blurring from misalignment. This is the type of hand-drawn characteristic that the Pen Project aims to capture.

³ These image sections are about 1/8th of the total image.

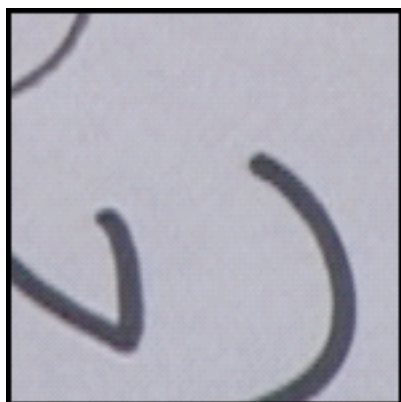


Image 5. Resulting section from a series of five zoomed in camera images

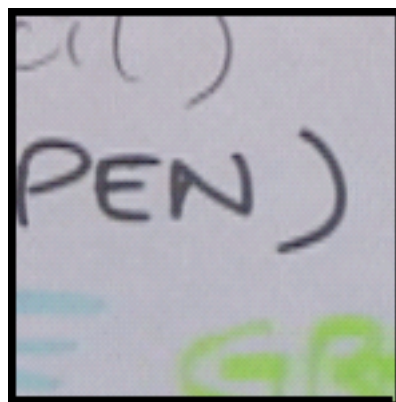


Image 6. Resulting section from another series of five zoomed out camera images

The two images above are the results of two separate image sequences (not shown) taken from the camera. The input sequences both had less camera movement than the first example. Because of this, they had very little scanline artifacts. They demonstrate what happens when a sharper marker is used. The result on the right also shows a variety of other markers.

I was very pleased with the final results I obtained. As seen above, the resulting images are very accurate. They do not lose much detail due to alignment errors. The results are quite consistent as well. Very rarely does an image get stuck in a local minimum that is not the true average pixel difference minimum. When this does occur, it can usually be attributed to input parameters that are too small. Tweaking these input parameters, n and k , correctly avoids any inconsistency. To put a number on consistency, my resulting image misaligned itself noticeably once out of about thirty trials.

After testing real data, I wanted to push the limits of my algorithm to determine any weaknesses. Thus, I hand drew a snake figure in a series of frames that moved across a page. The difference between my series and the real input was that I created a precise textured background. I was curious to see whether such a textured background would aid or detract from the accuracy of the algorithm.

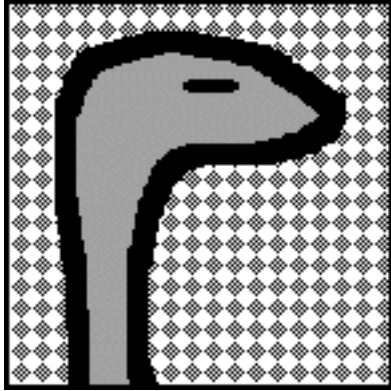


Image 6. One sample from the input series

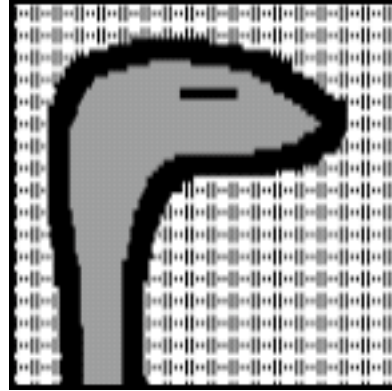


Image 7. A sample from the resulting image

As seen above, the snake head came out with some noticeable blurring and the background did not line up properly. The inaccurate background is most likely due to aliasing due to sampling. In addition, as is evident, the snake head is visibly blurred by the alignment.

DISCUSSION

Overall, the proposed method for image alignment seems very promising. It is obviously not complete, the times are not below the required threshold, but the results are remarkably accurate. There are three main areas for discussion: improving time, improving accuracy, and possible extensions to this project.

As mentioned earlier, there are many ways that this project could attain improved times. The first and probably most important method to improve performance would be to limit the comparison space. Rather than find the sum of differences squared over the whole image, one could easily pick only certain samples of the image to compute. These samples could be random, but better yet, could be based upon important features of the image such as edges or corners. In an environment like that of the Pen Project, which consists of mostly lines on paper, features would be a good way to determine alignment. Adding some sort of feature detection to the algorithm alongside pixel difference computation would make for a robust method at times where there are little or no detectable features on the page.

A second area for severe performance increase would be in the blending phase of the proposed method. Currently, four pictures are resampled to fit into the coordinate system of the center image. This extra step of bilinear interpolation could be avoided by blending directly from

the image space of each individual image. By combining resampling with blending,, the process would require much more math but would be faster than fully resampling and then blending.

Secondly, there are many things one could do to improve the accuracy of this project. Although the results indicate that this method aligns images very accurately, it still does not take into account rotation or any sort of projective transform. These are both ways in which to improve the accuracy of the method. However, as stated before, improved accuracy comes with the price of worse performance. Personally, I feel that the assumptions made for my method allow for fast enough processing time and result in sufficient accuracy.

Enhanced noise reduction is another way to improve the accuracy of this project. Currently, the only noise reduction method is to try to reduce noise through blending of multiple images. With this technique, random differences in pixel intensities due to camera noise can be balanced out by the sheer number of pixels averaged together for the output. However, this is not sufficient for full noise reduction. To truly obtain accurate results, additional noise reduction techniques would have to be used on the output image.

The main extension I see to this project, besides improving performance, would be trying to get increased resolution, or super-resolution, output from the input images. The concept behind this idea is that with so many input images, there are a lot of samples of any given area. With so many samples, one should be able to extract samples from the input at a higher resolution than that an individual input image. There are many papers that deal with exactly this problem, for example Freeman, Jones, and Pasztor[4]. Although most of these super-resolution techniques are computationally expensive, or require building up sample libraries, many of the ideas could be modified and applied to this project. Because capturing the fine detail of hand drawn artwork was a goal of the Pen Project, getting higher resolution and better detail as output would be an extremely beneficial addition.

CONCLUSION

In summary, the image registration method proposed in this paper serves as a fully functional way to register a series of images in real-time and deal with interlacing caused by cameras. Although the speed does not yet fall below the required threshold of the Pen Project, there are numerous areas in which to improve performance. The results obtained are equally accurate to

those of phase correlation and are consistent across varying input samples. There are many exciting applications of this method as well as some interesting extensions that would improve the detail obtained through registration.

REFERENCES

- [1]Brown, Lisa Gottesfeld. *A Survey of Image Registration Techniques*. ACM Computing Surveys, 24(4):325-376, December 1998.
- [2]Eberly, David. *3D Approximation source code* retrieved from the world wide web [<http://www.magic-software.com/Approximation3D.html>], May 2003.
- [3]Eberly, David. *Least Squares Fitting of Data*. [<http://www.magic-software.com/Documentation/LeastSquaresFitting.pdf>], September 2001.
- [4]Freeman, Jones, Pasztor. Example-Based Super-Resolution. *IEEE Computer Graphics and Applications*, v.22 n.2, p. 56-65, March/April 2002.
- [5]Galassi, Mark, et. al. *GNU Scientific Library: Reference Manual*. Ed. 1.3. [http://sources.redhat.com/gsl/ref/gsl-ref_toc.html]. December 2002.
- [6]Galassi, Mark, et. al. *GSL Library source code* retrieved from the world wide web [<ftp://ftp.gnu.org/gnu/gsl/gsl-1.3.tar.gz>]
- [7]Kuglin, C.D., and Hines, D. C.. The Phase Correlation Image Alignment Method. *Proceedings of the IEEE 1975 International Conference on Cybernetics and Society*, p. 163-165, New York, September 1975.
- [8] [<http://www.fft.org>], February 2003.