Efficient Global Weighted Least-Squares Translation Registration in the Frequency Domain

Jeff Orchard

University of Waterloo, Waterloo Ontario N2L 3G1, Canada,
jorchard@cs.uwaterloo.ca

Abstract. The weighted sum of squared differences cost function is often minimized to align two images with overlapping fields of view. If one image is shifted with respect to the other, the cost function can be written as a sum involving convolutions. This paper demonstrates that performing these convolutions in the frequency domain saves a significant amount of processing time when searching for a global optimum. In addition, the method is invariant under linear intensity mappings. Applications include medical imaging, remote sensing, fractal coding, and image photomosaics.

1 Introduction

One of the most common error metrics used in scientific applications is the sum of squared differences (SSD). In image processing, the SSD cost function is frequently used to asses the degree of similarity between two images. Image registration, in particular, often uses this metric when judging what spatial transformation brings two images of the same scene into alignment. It has been shown that for images differing only by additive Gaussian noise, the SSD cost function is the optimal choice [1]. Any problem that seeks to minimize the SSD is called a "least-squares" problem.

Another common error metric is cross-correlation [2]. One of the reasons for its popularity is the fact that its computation is equivalent to a convolution and can therefore be evaluated efficiently in the frequency domain (see section 2.1 below). Without this speedup, performing image registration would be too slow, particularly for 3D datasets or for large images (bigger than 1024×1024) such as those common in X-ray imaging and remote sensing. This method is common practice in medical image registration [3, 4].

In many image registration applications, only a small portion of each image is used to register the images. For example, one might have two overlapping aerial photographs, as in Fig. 1. If you can outline a window of the overlap in one of the photos, then finding the correct alignment of the two photos can be achieved by shifting one image over the other and evaluating the error norm in that window. The offset that gives the optimum norm value is called the optimal registration, and should correspond to the correct position. This windowed registration is

(a) Source Image (b) Target Image

Fig. 1. Overlapping aerial photographs. A region of the overlap is outlined in the target image.

equivalent to a weighted registration problem, where all the pixels outside the windowed region have a weight of zero.

It is sometimes necessary to cast the intensities of an image down to a scale that has a limited range. For example, suppose the intensity values of two overlapping images have to be mapped to the range $[0, 255]$. The way the cast is typically done is to create a linear map such that the lowest intensity in the image maps to 0, and highest to 255. If the intensity ranges of two overlapping images is different, the intensity remappings will be different. This intensity casting causes corresponding pixels in the two images to have different intensities. The mismatch confuses the cross-correlation and SSD cost functions, and can lead to an incorrect registration result.

This paper proposes a method to efficiently compute the weighted SSD cost function by representing it as a combination of convolutions. Also, the optimal linear intensity remapping is determined with little additional effort.

2 Theory

2.1 Correlation Coefficient

When it comes to comparing images or functions, a common metric to measure the degree to which functions are similar is the Pearson's cross-correlation, defined as [2]

$$C(a) = \frac{\int f(x)g(x-a)dx}{\sqrt{\int f^2(x)dx \int g^2(x)dx}} \,. \tag{1}$$

In this context, $C(a)$ measures the correlation between the function $f(x)$ and the shifted function $g(x-a)$. For example, if g is equal to f, then $C(a)$ achieves its maximum value when a is zero (corresponding to no shift). This measure has been used in automatic alignment algorithms in medical imaging [3, 5, 6].

It is well known that the operation of convolution can be carried out by a multiplication in the frequency domain. The convolution of two functions, $f(x)$ and $g(x)$, is defined as

$$(f \star g)(a) = \int_{-\infty}^{\infty} f(x)g(a-x)dx \ . \tag{2}$$

That is, the function g is flipped about $x = 0$ and shifted along the negative x-axis by a distance a. To turn the numerator of (1) into a convolution, we define a new function $\bar{g}(x)$ that is equal to $g(-x)$. Then, we replace $g(x-a)$ in (1) with $\bar{g}(a-x)$. Now the numerator of (1) is a convolution between $f(x)$ and $\bar{g}(x)$.

Consider the Fourier transform of the convolution, $\mathcal{F}\{(f \star \bar{g})(a)\}$. It is not difficult to show that the Fourier transform of a convolution is equivalent to the component-wise product of the two Fourier transforms (see Appendix B of [7]). That is, $\mathcal{F}\{(f \star \bar{g})(a)\} = \mathcal{F}\{f(x)\}(s) \cdot \mathcal{F}\big\{\bar{g}(a-x)\big\}(s)$. Thus, the numerator of the cross-correlation function can be evaluated for all values of a by taking the Fourier transform of each of f and $\bar{g}$, multiplying the two sets of coefficients, and then applying the inverse Fourier transform to the result. Finding the maximum of this function with respect to a is simply a matter of scanning the evaluated function $C(a)$ for its maximum value. Note that if f and g are real-valued, then $C(a)$ will also be real-valued.

One of the problems with the correlation coefficient cost function is that it cannot be used as a measure for weighted registration problems. For weighted registration problems, we turn to the sum of squared differences (SSD) cost function.

2.2 Weighted Sum of Squared Differences

Given the functions $f(x)$ and $g(x-a)$ as before, the weighted sum of squared differences (SSD) registration of g to f, with weighting function w, corresponds to the value of a that minimizes

$$L_W(a) = \int \big[f(x) - g(x-a)\big]^2 w(x-a)dx \ . \tag{3}$$

The weighting function, w, is greater than or equal to zero over its entire domain. For example, w could be a piecewise constant function that is zero everywhere except in a region where the registration is to operate – there its value is 1. Then, minimizing (3) is the same as moving w in concert with g, and limiting the domain of the SSD calculation to only the region where w is non-zero. By expanding the square brackets in (3), we get

$$\begin{aligned} L_W(a) = & \int f^2(x)\bar{w}(a-x)dx + \int \bar{g}^2(a-x)\bar{w}(a-x)dx \\ & -2\int f(x)\bar{g}(a-x)\bar{w}(a-x)dx \ , \end{aligned} \tag{4}$$

where $\bar{g}$ is defined as before, and $\bar{w}$ is defined similarly. As with the evaluation of the cross-correlation function, the weighted SSD cost function includes

convolution-like terms. The first integral is a convolution between f^2 and $\bar{w}$, and changes with different values of a. We will denote it $e_1(a)$. The second integral is a constant with respect to a since the weighting function moves with $\bar{g}$. We will denote this value as e_2. Combining $\bar{g}$ and $\bar{w}$ so that their product $\bar{g}(x)\bar{w}(x)$ equals $\bar{h}(x)$, the last integral of (4) becomes the convolution $\int f(x)\bar{h}(a-x)dx$. We will denote this last integral as $e_3(a)$. Thus, the weighted SSD cost function for a given displacement a is

$$L_W(a) = e_1(a) + e_2 - 2e_3(a) \ . \tag{5}$$

2.3 Intensity Remapping

In addition to finding the best match over all shifts, we can also find the best match over all linear intensity remappings. That is, we wish to find the optimal shift in conjunction with the optimal contrast and brightness adjustment to make the corresponding parts of the two images as similar as possible. This is analogous to replacing the intensity $g(x)$ with $sg(x) + t$ for some constants s and t. Naturally, the optimal s and t will depend on the shift, a. With an intensity-remapped g, the weighted SSD error measure can be written

$$L_R(a,s,t) = \int \left[f(x) - s\ g(x-a) - t\right]^2 w(x-a)dx \ . \tag{6}$$

Now the problem becomes a minimization over a, s and t. In particular, for every value of a, we wish to find the corresponding optimal values for s and t. The optimal values can still be found efficiently since the convolution integrals that arise can still be evaluated in the frequency domain. Writing F, G and W instead of $f(x)$, $g(x-a)$ and $w(x-a)$, respectively, we expand the brackets in (6) to get,

$$L_R(a,s,t) = \int F^2W + s^2G^2W - 2sFGW + t^2W - 2tFW - 2tsGW dx \tag{7}$$

$$= \ e_1(a) + s^2e_2 - 2se_3(a) + t^2e_4 - 2te_5(a) + 2tse_6 \ . \tag{8}$$

Notice that (8) implies that e_1, e_3 and e_5 are functions of a, while e_2, e_4 and e_6 are constants (since g and w shift with each other). Only the integrals that involve f with g or w change with respect to a. For any given value of a, (8) is a paraboloid in s and t that opens upward. The minimum value of the paraboloid can be determined analytically by solving a simple 2×2 linear system of equations. Hence, for a fixed a-value, the optimal s- and t-values are given by

$$\begin{bmatrix} s \\ t \end{bmatrix} = \frac{1}{e_2e_4 - e_6^2} \begin{bmatrix} e_4 & -e_6 \\ -e_6 & e_2 \end{bmatrix} \begin{bmatrix} e_3 \\ e_5 \end{bmatrix} . \tag{9}$$

Although the theory derived here is for 1D functions, it can easily be generalized to higher dimensions. For the remainder of this paper, we will focus on 2D images.

2.4 Algorithmic Complexity

All the above analysis involving the Fourier transform also carries over to the discrete Fourier transform (DFT). The discrete 2D analog for (3) is

$$L_W(a) = \sum_{i=1}^{N} \sum_{j=1}^{N} \left[f_{i,j} - g_{i-a,j-b} \right]^2 w_{i-a,j-b} \ . \tag{10}$$

In this section, we compare the cost (measured in floating point operations, or flops) of computing the optimal solution for (10) by the direct computation method, to the cost of evaluating (10) by performing convolution in the frequency domain.

Suppose f and g are $N \times N$ images. The direct method to evaluate (10) involves simply evaluating the double-sum for every valid shift (a, b). For a single value of (a, b), evaluating the double-sum requires adding together N^2 terms, and each term requires one subtraction and two multiplications (since squaring is a multiplication). Thus, at 3 flops per term, the double-sum takes $(3N^2 + N^2 - 1)$ flops to evaluate. Since there are N^2 values of (a, b), evaluating (10) for all values of (a, b) takes $(4N^4 - N^2)$ flops.

However, evaluating (10) by calculating the convolutions in the frequency domain (via the form in (4)) takes $\mathcal{O}(N^2 \log N)$ flops. This is because the Fast Fourier Transform (FFT) of an $N \times N$ image takes at most $4N^2 \log_2 N$ complex operations (a complex multiplication followed by a complex addition) [8]. Each complex operation requires 8 flops, so the FFT takes at most $32N \log_2 N$ flops. To find the minimum of (4), a total of 5 FFTs need to be performed: FFTs of f, f^2, $\bar{w}$, $\bar{h}$, and an inverse FFT to transform the measure back to the spatial domain. This brings the total number of flops to perform the FFTs to $160N^2 \log_2 N$. Other than the FFTs, the remaining tasks are all $\mathcal{O}(N^2)$. These tasks include evaluating the middle term in (4), and performing the element-by-element multiplication of the Fourier transforms.

In many applications, the weighting function w is zero for a large portion of the domain. To analyze this situation, assume that w is non-zero over a domain of size $M \times M$, where $M < N$. Then, the sum in (10) has only M^2 terms, and hence the cost of evaluating it directly for a single value of a is $(3M^2 + M^2 - 1)$ flops, and the cost of evaluating it directly for all values of a is $(4N^2M^2 - N^2)$ flops, or $\mathcal{O}(N^2M^2)$.

Unfortunately, the Fourier method is not any cheaper to evaluate when $M < N$; the cost is the same as if w were nonzero everywhere. However, the Fourier method is still cheaper than the direct method if $M^2 > \log N$.

The above complexity analysis is for the simple weighted SSD cost function that does not include any intensity remapping. However, similar results are obtained for the more complicated intensity remapping method. In (8), the terms $e_1(a)$, $e_3(a)$ and $e_5(a)$ all involve convolutions.

3 Methods

We implemented both the direct method and the Fourier method in the C++ programming language. All the Fourier transforms were done using the FFTW library [9]. Image data was stored in contiguous memory to improve the cache coherency (i.e. to cut down on the number of cache misses). The direct method evaluates the error norm for only those shifts that have the entire window (non-zero part of w) completely inside f.

On a set of satellite images from Intermap Technologies Inc. (Englewood, Colorado), we timed how long each method took to find the optimal shift and intensity remapping parameters. The timings were run on a 2.4 GHz Intel Pentium 4 machine with 2 gigabytes of RAM.

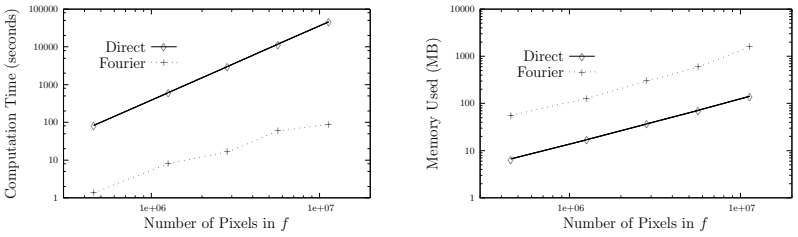
The images were shrunken to various sizes to get a more complete picture of their performance on different scales. Figure 1 shows the two images that were used, and the window for comparison. For the largest set of images, f had dimensions 3008×3780, g and w had dimensions 3078×3845, and the window had dimensions 490×2460. The four scaled-down sets of images had roughly $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{9}$ and $\frac{1}{25}$ the number of pixels in each image.

4 Results and Discussion

In all tests, both methods successfully determined the optimal translation and intensity remapping parameters. Figure 2 shows the absolute difference image of the two images merged using the optimal shift. The region of overlap is nearly zero, indicating that the match is excellent.

The timing results are summarized in Fig. 3(a). The figure is a log-log plot graphing the number of pixels in f (the source image) against the running time in seconds. Note that f, g and w were all scaled equally for each execution. The straight line of the direct method indicates that there is a power-law relationship between the scale of the problem (in terms of number of pixels in f) and the running time. The fact that the slope of the line is approximately 2 (with respect to the logarithm of the axis labels) fits with the algorithmic complexity derived earlier. In particular, it says that the computation time is proportional to the square of the number of pixels in f (where f is $N \times N$).

Fig. 2. Difference image of registered images.



(a) Computation time (log-log) (b) Peak memory usage (log-log)

Fig. 3. Computation time and peak memory usage for the direct method and the Fourier method. For each run, each of the images f, g and w were resized using the same scale factor. Note that all axes are plotted in log scale.

The graph for the Fourier method is not as easy to interpret. However, the graph is consistent with the complexity class derived above: $\mathcal{O}(N^2 \log N)$.

To get a better feel for the meaning of Fig. 3(a), let us consider some example timings. For the smallest dataset, in which f and g are roughly 615×760 and the window is 96×480, the Fourier method took 1.36 seconds and the direct method took 83.7 seconds. For the largest images, in which f and g are roughly 3008×3780 and the window is 490×2460, the Fourier method took 88 seconds and the direct method took 46,873 seconds (just over 7 hours).

It should be noted that the prime factorization of the dimensions of f play a role in the speed of the Fourier method. The FFT is a divide-and-conquer algorithm and is most efficient when the length of the data can be factored into a product of small prime numbers. The above experiments represent a rather optimistic scenario, in which the dimensions of f have lots of small prime factors: $3008 = (2)^6(47)$, and $3780 = (2)^2(3)^3(5)(7)$. However, the slow-down is not terribly significant for less fortunate image dimensions. If f has dimensions 3019×3796 (3019 is a prime number, and $3296 = (2)^2(13)(73)$), the Fourier method takes 168 seconds, slower by a factor of approximately 2.

The memory use by the two methods is also quite different. Figure 3(b) is a log-log plot showing the peak memory usage (in megabytes), again with the number of pixels in f on the horizontal axis. Both methods show a straight line with a slope of roughly 1 (with respect to the log of the axis labels). Hence, as we might expect, the memory requirements go up linearly with the number of pixels. However, the Fourier method used about eight times as much memory as the direct method (using over 1.5 gigabytes to process the largest image set). The reason for this is that the images have to be stored as complex numbers (single-precision). Another reason is that the Fourier method has to compute and store $e1$, $e3$ and $e5$ in their entirety before evaluating $L_R(a, s, t)$. The direct method can calculate these values one trial shift at a time. Furthermore, the direct method does not need to store all of g and w, only the parts corresponding to where w is non-zero. Our implementation takes advantage of this shortcut.

5 Conclusions and Future Work

When registering two images that are translated with respect to each other, the SSD cost function involves a convolution term. Variants of the problem, including the addition of a weighting factor and linear intensity remapping, still yield convolution terms. The computational advantage of evaluating these terms in the frequency domain is very substantial. In our experiments, the Fourier method was at least 60 times faster (in some cases over 500 times faster) than the direct method. The trade-off is the mount of memory required by the methods; the Fourier method used about eight time as much memory as the direct method.

In most imaging applications, the original data to be aligned is real-valued (i.e. the imaginary part is zero). For real-valued data, the FFT can be done faster by taking advantage of the symmetry in the Fourier coefficients. Indeed, the FFTW library has methods to compute the FFT of a real-valued dataset, outputting a half-size set of Fourier coefficients (avoiding the redundancy caused by the symmetry). Adapting the Fourier method described in this paper to take advantage of this efficiency is trivial.

It should be noted that the direct method has some advantages. For example, the cost function can be evaluated for a subset of trial shifts, while the Fourier method is an inherently global operation. Thus, if the approximate registration is known, it might be more effective to directly evaluate the cost function for sample shifts around that initial guess rather than performing a global search using the Fourier method. However, for the direct method to be faster on the largest image set, fewer than 0.2% of the possible shifts could be sampled. More than that and the Fourier method would be faster.

The SSD error measure is not necessarily the best cost function for registering images. Which error norm is best will depend on a number of factors, such as the type of noise present in the images. Some examples of other norms are the $\mathcal{L}^1$ norm, total variation [10], and robust estimators [11, 12]. It may be feasible to expand these error norms using a Taylor series. The convolution terms in the Taylor series could then be evaluated in the frequency domain. The problem is that more terms makes the Fourier method more expensive, and it is not clear which error norms will still be faster using this approach. More work to investigate other norms is needed.

Acknowledgment

We would like to thank Intermap Technologies Inc. for supplying the satellite image dataset. We also thank the Natural Science and Engineering Research Council (NSERC) of Canada for their financial support.

References

1. Hill, D.L.G., Hawkes, D.J.: Across-modality registration using intensity-based cost functions. In Bankman, I., ed.: Handbook of Medical Imaging: Processing and Analysis. Academic Press (2000) 537–553

2. Woods, R.P.: Within-modality registration using intensity-based cost functions. In Bankman, I.N., ed.: Handbook of Medical Imaging: Processing and Analysis. Academic Press (2000) 529–536
3. Maas, L.C., Frederick, B.D., Renshaw, P.F.: Decoupled automated rotational and traslational registration for fMRI time series data: the DART registration algorithm. Magnetic Resonance in Medicine **37** (1997) 131–139
4. Pipe, J.G.: Motion correction with PROPELLER MRI: Application to head motion and free-breathing cardiac imaging. Magnetic Resonance in Medicine **42** (1999) 963–969
5. Ehman, R.L., Felmlee, J.P.: Adaptive technique for high-definition MR imaging of moving structures. Radiology **173** (1989) 255–263
6. Wang, Y., Grimm, R., Felmlee, J., Riederer, S., Ehman, R.: Algorithms for extracting motion information from navigator echoes. Magnetic Resonance in Medicine **36** (1996) 117–123
7. Orchard, J.: Simultaneous Registration and Activation Detection: Overcoming Activation-Induced Registration Errors in Functional MRI. PhD thesis, Simon Fraser University (2003)
8. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex fourier series. Mathematics of Computation **19** (1965) 297–301
9. Frigo, M., Johnson, S.G.: The design and implementation of FFTW3. Proceedings of the IEEE, Special Issue on Program Generation, Optimization, and Platform Adaptation **93** (2005) 216–231
10. Li, Y., Santosa, F.: A computational algorithm for minimizing total variation in image restoration. IEEE Transactions on Image Processing **5** (1994) 987–995
11. Nestares, O., Heeger, D.J.: Robust multiresolution alignment of MRI brain volumes. Magnetic Resonance in Medicine **43** (2000) 705–715
12. Nikou, C., Heitz, F., Armspach, J.P., Namer, I.J., Grucker, D.: Registration of MR/MR and MR/SPECT brain images by fast stochastic optimization of robust voxel similarity measures. NeuroImage **8** (1998) 30–43