Published in *Proceedings of the SPIE Astronomical Telescopes and Instrumentation Symposium: New Frontiers in Stellar Interferometry*, Glasgow, Scotland, June, 2004. (SPIE vol. 5491-173).

Adaptive DFT-based fringe tracking and prediction at IOTA

Edward Wilson[*a], Ettore Pedretti[bc], Jesse Bregman[d], Robert W. Mah[d], Wesley A. Traub[b]

[a]Intellization, 454 Barkentine Ln, Redwood Shores, CA 94065-1126
[b]Harvard-Smithsonian Center for Astrophysics, 60 Garden Street, Cambridge, MA 02138
[c]University of Michigan, Astronomy Dept, 914 Dennison Building, Ann Arbor MI 48109
[d]NASA Ames Research Center, MS 269-1, Moffett Field, CA 94035

ABSTRACT

An automatic fringe tracking system has been developed and implemented at the Infrared Optical Telescope Array (IOTA). In testing during May 2002, the system successfully minimized the optical path differences (OPDs) for all three baselines at IOTA. Based on sliding window discrete Fourier transform (DFT) calculations that were optimized for computational efficiency and robustness to atmospheric disturbances, the algorithm has also been tested extensively on off-line data. Implemented in ANSI C on the 266 MHz PowerPC processor running the VxWorks real-time operating system, the algorithm runs in approximately 2.0 milliseconds per scan (including all three interferograms), using the science camera and piezo scanners to measure and correct the OPDs. Preliminary analysis on an extension of this algorithm indicates a potential for predictive tracking, although at present, real-time implementation of this extension would require significantly more computational capacity.

Keywords: Interferometry, fringe tracking, IOTA, discrete Fourier transform, interferogram, predictive control

1. INTRODUCTION

The Infrared Optical Telescope Array (IOTA), shown in Figure 1, is a 3-aperture long baseline Michelson stellar interferometer located on Mt. Hopkins near Tucson, Arizona. Three 45-cm collectors can be located along a 15 m by 35 m L-shaped array, supplying visible and near-IR light to pupil-plane beam combiners. The operational details and scientific accomplishments of IOTA have been well documented by other authors[1,2,3].

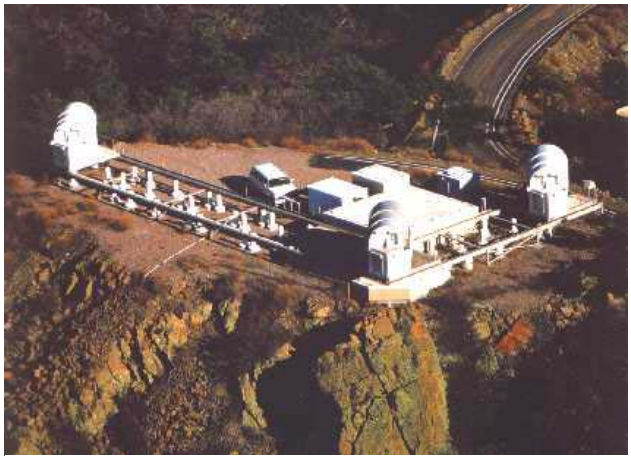

Figure 1: Infrared-Optical Telescope Array (IOTA)

This article reports on the development of algorithms designed and used to simultaneously null the optical path differences (OPDs) for the three baselines provided by IOTA's three apertures. Examples of fringes with relatively higher (on the left) and low (on the right) signal-to-noise ratios (SNRs) are shown below. These scans were taken on the same delay line, about 2 seconds apart. The relative confidence in the fringe center identification is indicated as the number on the plot, and will be discussed later. Significant sources of noise include atmospheric turbulence, vibration, photon noise, and detector noise. The goal of the fringe tracking system is to perform coherencing (vs. cophasing), by controlling the OPD to allow the interferogram to be captured in the presence of bad seeing conditions, and fainter objects. The controller works by identifying the fringe-center locations on all 3 interferograms following each scan, and then adjusting the centers-of-travel of the piezo-driven scanning mirrors, attempting to keep the fringe packets centered in all 3 scan windows.

[*] Ed.Wilson@intellization.com; phone 1 650 604-0465; fax 1 650 604-3594

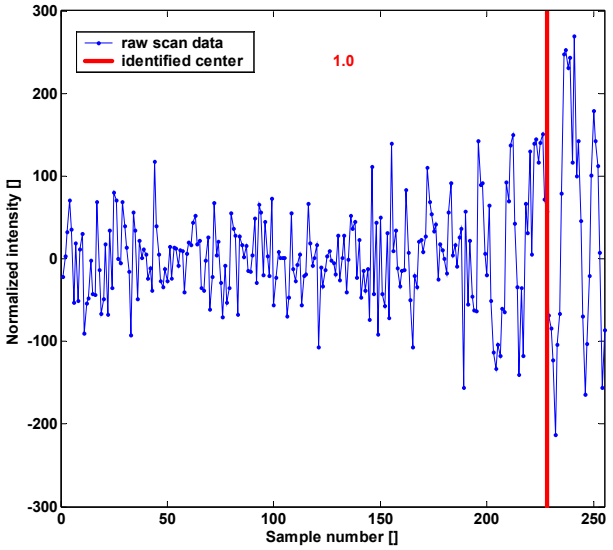

Figure 2: typical IOTA scan with low SNR

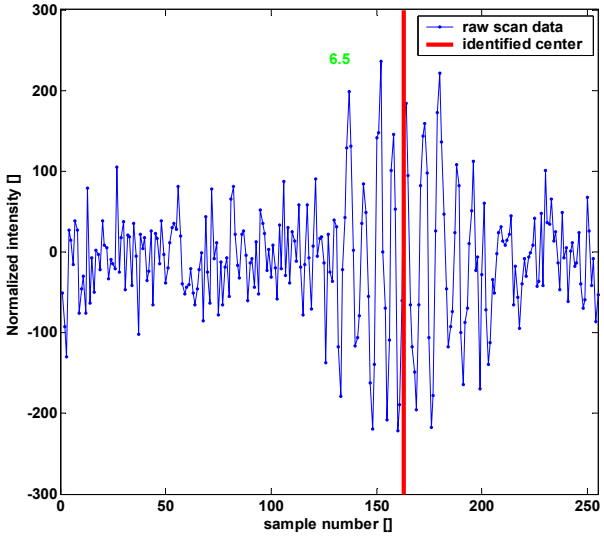

Figure 3: typical IOTA scan with higher SNR

Details of the relevant interferometric derivations is covered thoroughly by other authors[4,5]. The idealized fringe packet function is a sinc function multiplied by a sinusoid, and can be represented with the following equation, where y is the normalized intensity, x is the sample number, and A, B, C, D, E are parameters defining respectively the amplitude, sinc-function width, sinc function center, sinusoid (fringe) frequency, and sinusoid phase shift. Many equivalent variants on this functional form are of course possible (for example, substituting $\sin(dx+e)$ for $\cos(D(x+E))$); this one was chosen to facilitate the particular gradient-based optimization procedure developed. Although E is not an independent variable in theory, in practice, noise and atmospheric disturbances make it independent of the other parameters.

$$y = A\,\mathrm{sinc}(B(x+C))\cos(D(x+E)) \tag{1}$$

Figure 4 shows an idealized packet with parameters chosen to approximate that from the higher-SNR real-data fringe shown previously. The computing and actuation aspects of the control system are described by Traub and Pedretti[2,5]; the present article details the fringe tracking algorithm and aspects of its software implementation.

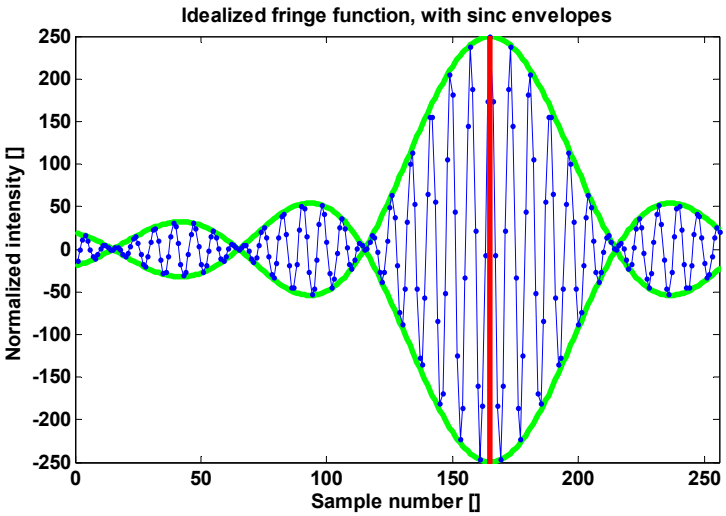

Figure 4: Idealized interferogram with sinc envelopes

Due to the significant noise sources present, the fringe tracking algorithms developed here were developed through extensive testing on actual data sets from IOTA, dating back to 1997 (as opposed to working with simulated data). The algorithms were developed with autonomous adaptability (due to widely varying seeing conditions and object intensities), robustness (absolute accuracy is not as important as keeping the fringe within the scan window), and computational efficiency (requiring a minimal amount of computation time due to the limited resources and need for fast scanning – typically 3 Hz).

1.1. Related research

Interferometric fringe tracking is a broad field, so this section will focus on fringe tracking developments at IOTA.

Wilson developed a method, summarized in this paper, that used the envelope of the interferogram to identify the packet center, and a gradient-based optimization method for refinement of this estimate[6]. Although fast and robust, it did not make use of the fringe frequency, leading to the present research, which makes this improvement. This was an off-line study using IOTA data taken in 1997.

Morel and others on the IOTA team worked to implement the core aspect of Wilson's 1999 algorithm on the IOTA scanning hardware[7]. The fringe-center identification aspect of the system was found to be very robust and accurate even with very noisy signals, but the slow response of the control communications and actuation hardware made the overall control system ineffective. The control computing, communications, and actuation hardware was subsequently upgraded to permit further implementation efforts[2].

Pedretti developed a fringe tracking algorithm taking a completely different approach, based on double Fourier Interferometry (DFI)[5,8]. This method calculates the group delay of fringes dispersed with DFI, which is used to obtain the wavelength dependent phase from the fringe packet. This method has also been implemented at IOTA on the current hardware, and is used there regularly. A performance comparison of the different approaches at IOTA is presently underway.

Thureau developed a fringe envelope tracking algorithm at COAST, which was subsequently implemented for testing at IOTA[9].

1.2. Approach

Guided by a background in signal processing and system identification (ID), the original approach taken towards fringe tracking was to fit the parameters in Equation 1 to the data on each scan, with the fringe center then contained in C. A nonlinear, gradient-based optimization was developed to perform this, with extensive testing and tuning on representative IOTA data sets from 1997. This nonlinear optimization required a reasonably close initial estimate for C, which was provided by processing the fringe packet envelope. As it turned out, the accuracy of this initial estimate was generally within a sample or two (out of 256 points in a scan, typically) of the result following the full nonlinear ID. Given implementation constraints and the existence of other more significant error sources, it was decided that this initial estimate processing could serve as the on-line fringe ID algorithm. This was tested on-line in 1999 and 2000[7].

In 2002, following the instrument control hardware upgrades and in preparation for a second implementation attempt, the algorithm was updated. The original envelope-based algorithm basically drew an envelope around the data and found the hump, thereby completely ignoring the fringe frequency, D. As can be seen in the example data given previously, the fringe frequency is visible in the fringe packet, and is relatively obscured by noise outside the center (although it is still there). The improvement looks for intensity amplitude at the fringe frequency, rather than at all frequencies (as the envelope-based ID did). This is accomplished with an efficiently implemented sliding window discrete Fourier transform (DFT). This updated algorithm was implemented in February 2002 at IOTA, with testing on simulated fringes through the instrument, and later on-the-sky testing with all 3 apertures in May 2002. Being more physically based, the change was made with the expectation that it would be more robust for future data and algorithm changes.

1.3. Article overview

The DFT- and envelope-based tracking algorithms, both of which are concise enough to implement on the real-time system, are presented in Sections 2 and 3. A gradient-based optimization algorithm to identify all fringe packet parameters (A, B, C, D, E) from Equation 1 is developed in Section 4. Initial proof-of-concept results for using these identification results for fringe motion prediction are presented in Section 5. Implementation issues and conclusions are presented in Sections 6, and 7.

2. DFT-BASED TRACKING

A more detailed description of the algorithm specifics is described elsewhere by Wilson[10].

2.1. Algorithm summary

1. A window (nominally of a length containing two fringe periods, but can be set to any integer) is passed over the data, where a single-frequency discrete Fourier transform (DFT) is calculated to try to detect the expected fringe frequency (this frequency is adaptively updated – by changing the window size - after each scan). The DFT is calculated 5 times for each scan, using window sizes of nominal plus [-4, -2, 0, +2, +4]. The number of points in the window is odd so the center lands on a point. The relative scaled magnitudes of these DFT results are used to determine the nominal window size for the next scan.
2. Each of the 5 DFT results is smoothed using a rectangular averaging filter.
3. The point-by-point maximum of the 5 smoothed DFT results, referred to as the composite DFT result, is taken for further processing. Steps 1, 2, and 3 make this result more robust to intra-packet fringe frequency variations than a single-frequency DFT scan would be. The frequency corresponding to the largest DFT magnitude is chosen as the nominal frequency for the following scan – providing adaptive response to changing interferogram properties, and eliminating the need to initially set this carefully.
4. A fringe-packet-finding template is convolved with the composite DFT result, providing a peak when the composite DFT result matches the template shape. For computational efficiency, a rectangular template is used in place of a sinc-shaped template.
5. The sample corresponding to the maximum value of the previous step is used as the identified fringe-packet center.
6. A confidence metric is calculated based on the relative magnitudes of the composite DFT result near the ID'ed center and the background.
7. The previous steps are performed on all aperture pairs (3 in the case of IOTA), and the ID results and corresponding confidence metrics are combined to determine the scan centers for the next scan (to begin within a couple of milliseconds).

2.2. Algorithm steps illustrated on example data

The algorithm steps are presented using actual data, as shown in the following figures. They were generated using data collected from the AB-fringe of the 6th scan of the iota4 dataset on April 28, 2002; targeting star 12Iot_Dra; RA (J2000): 15.415278; Dec (J2000): 58.966111. Figure 5 shows the normalized raw data, as well as the result of the center identification that came after all steps were completed.

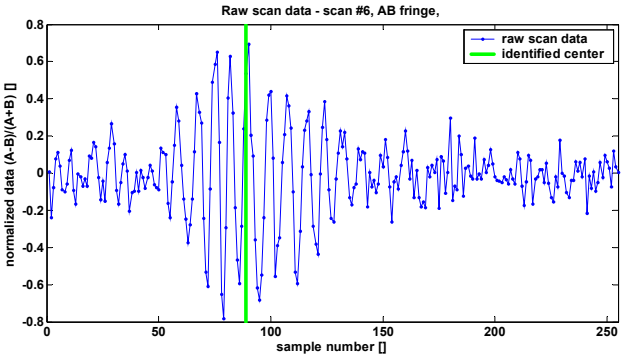

Figure 5: Normalized raw data

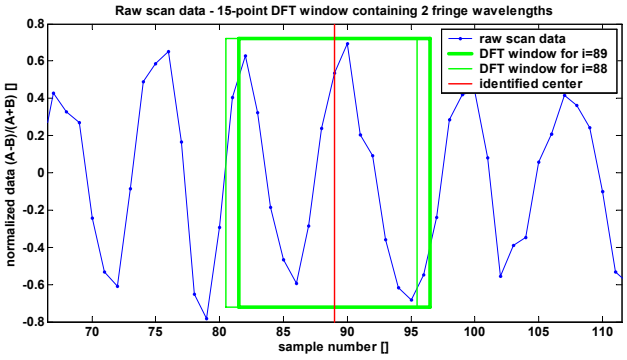

Figure 6: DFT sliding window

Figure 6 shows how the DFT window is passed over the raw data. The purpose of the DFT is to locate areas in the scan where the expected fringe frequency is present. A few things are done to greatly improve the efficiency of the DFT calculation – note it is not calculated as an FFT. This DFT calculates the magnitude of the signal in only one frequency bin – that corresponding to the fringe frequency. Also, a rectangular window is used, which enables very fast computation as the window is passed over the data. Calculating each new data point requires adding a term for the

incoming sample and subtracting a term for the leaving sample. So for example, to calculate the DFT for the fringe frequency centered at sample #89 in the figure uses the DFT result for sample #88, then adds a term for point #97 and subtracts a term for point #81.

The DFT window size is chosen in this case to nominally contain exactly two fringe wavelengths. Two wavelengths appears to be a good compromise between accuracy on clean scans (with higher coherence time, more accuracy would be possible with a larger window, but on clean scans tracking accuracy is not difficult) and noisy scans (if the coherence time is much below two wavelengths, there are probably no fringes to be seen).

For calculation of the DFT over the full spectrum, the first component ("bin") would correspond to the overall bias, the second component would correspond to a full wavelength extending across the full window, and the third component would correspond to two full wavelengths. Since the window size was chosen to cover two full fringe wavelengths, we calculate only the third component of the full-spectrum DFT. The real and imaginary parts are computed and then combined to produce the magnitude. The phase information is not used. This result is then smoothed using a sliding window having the same width as the DFT window (two wavelengths in this case), with the mean over the window producing the result shown in Figure 7. This step is computationally efficient, and reduces the variability in the DFT results. Even though it is calculated very differently, the result is very similar to that resulting from the envelope-finding calculations described in Section 3. Although the envelope-finding calculations provided excellent results, this DFT calculation is more physically based and is expected to provide better robustness for noisy signals.

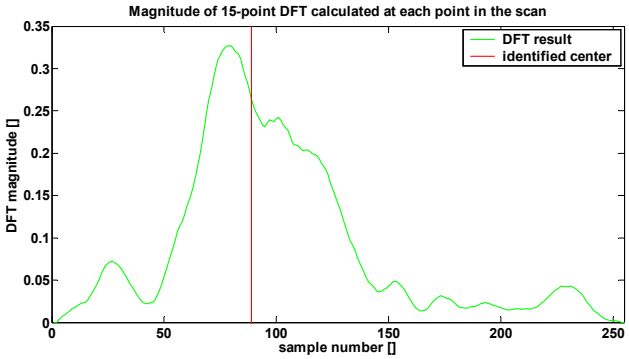

Figure 7: Smoothed DFT result

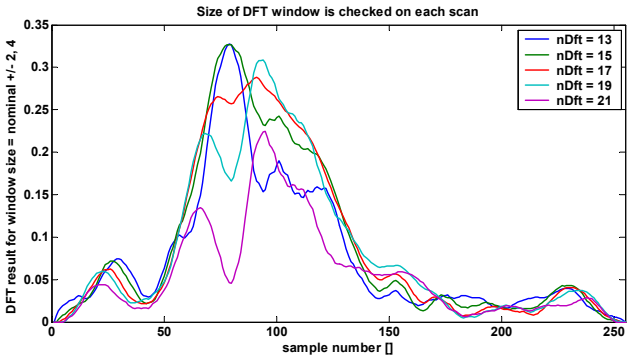

Figure 8: Multiple smoothed DFT results

Figure 8 summarizes the results of the smoothed DFT calculations using five different window sizes. Since the fringe frequency is not known exactly, and may change, this algorithm adapts to use the best window size. For every scan, the smoothed DFT calculations discussed above run five times: once for the window size used on the previous scan, once each for 2 and 4 samples larger, and once each for 2 and 4 smaller. The window size (nDft, as shown in the plot legend, is the number of samples) corresponding to the maximum smoothed DFT value (in this case, the highest DFT result occurs for nDft = 15 at sample #79) is chosen and carried through to the next scan as the nominal window size. To prevent nDft from increasing or decreasing too rapidly during periods of low signal-to-noise ratio, only one step up or down is permitted per scan.

The "composite DFT result" is formed by taking a point-by-point maximum over the 5 smoothed DFT results. This is the vector that is passed on for further processing. The smoothed DFT results (shown in Figure 8) are compensated based on the window size to allow meaningful comparison among the different results at each point. As can be seen by the 5 individual curves here, the composite DFT result is more representative of the fringe packet than any single smoothed DFT result would be. This is due to the intra-packet variations in fringe frequency, caused primarily by atmospheric distortion.

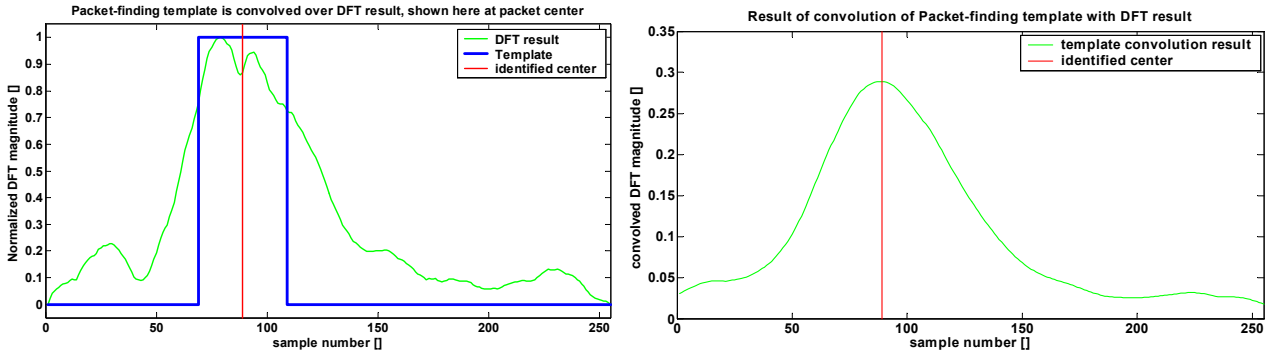

Figure 9: Composite DFT result before (left) and after (right) convolution with packet-finding window

Figure 9 illustrates how a template is convolved with the composite DFT result. The template is very simple, composed of ones and zeros, leading to very efficient computation: additions and subtractions at the template transitions only, as the template is passed over the composite DFT result – as opposed to an arbitrary convolution, which would require multiplications across the full template, repeated when centered at each point in the scan window. The template is very loosely modeled after what the ideal DFT result should look like (abs(sinc)-like). The computation is efficient, since after the initial computation for the first sample, each additional sample calculation involves only one add and one subtract (no multiplies or divides) – corresponding to the two vertical edges on the template. The +1 region is chosen to correspond approximately to the width at half the composite DFT result. The software implementation allows this width to be set easily, and an extension to make it adaptive should be feasible, although performance appears to be very robust to this number. For example, a value of nFringe = 35 for the half-width (meaning the template spans 71 samples) was used to successfully track simulated fringes at IOTA with the scan set to both 30 and 15 microns (the 15-micron scan had a fringe packet twice as wide as that of the 30-micron scan). The index corresponding to the maximum value of this result is used as the packet center estimate.

2.3. Confidence metric calculation

Once the fringe packet center has been identified, a decision must be made as to the result's level of validity of and degree to which it should be used to update the scan-center position. In cases where the fringe packet disappears momentarily, it is better to do nothing (keep scanning in the same location) than to chase the noise.

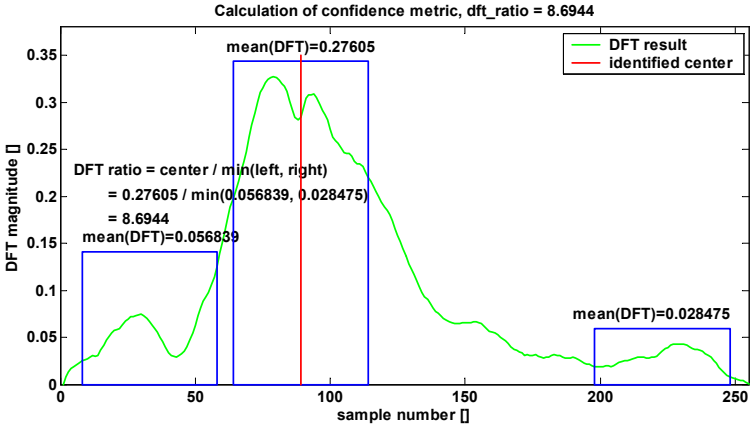

Figure 10: DFT-based confidence metric

This calculation is shown graphically in Figure 10. The concept is that the DFT calculation near the identified center should have a measurably higher value than the DFT calculation on the background noise. The mean of the DFT in windows spanning 20% of the scan width is calculated at the left edge, right edge, and at the identified center – as shown by the blue rectangles in the figure. The ratio of the mean DFT at the identified center to the smaller of the two edge measurements (minus one) is taken as the confidence metric. The reason to take both edges and then use the minimum is that this will give a valid background measurement even if the fringe falls at the edge of the scan.

One approach to using this confidence metric would be discrete: set a confidence threshold and either use or ignore the result based on that comparison. Setting the threshold value will depend on the level of tracking accuracy desired, the relative scan width, and other factors. It is a balance between the cost of accepting a wrong estimate and ignoring a valid

one – these costs vary depending on the application. Some additional complications with this method are raised when considering multi-aperture Interferometry, where changing the scan center on a single delay line affects two interferograms.

2.4. Three-scan tracking

When the third telescope at IOTA went on-line in February 2002 and 3-scans became available for tracking, the approach taken was to try to estimate the fringe center (related to OPD) for each of the three scans independently (as if only two telescopes were present), then to combine these results, along with the corresponding confidence estimate for each result, into the updated scan centers for the next scan. Ettore Pedretti at IOTA had already developed an algorithm to optimally merge the individual fringe center IDs, along with corresponding confidence estimates[5]. This algorithm was used directly.

2.5. Optional operations

Non-rectangular template – If the approximate width of the sinc function is known, a template of such a shape could be used in place of the rectangular template used in Section 2.5. A simpler version that is far more computationally efficient than a sinc-shaped template, and only slightly less efficient than the rectangular template, would have 3 rectangular regions as shown in Figure 11. This was used as part of the implementation in February 2002, but was later changed (to the rectangular) since distortion due to edge effects was found to occur more frequently than expected due to the narrower scan width. However, it should be considered a viable option.

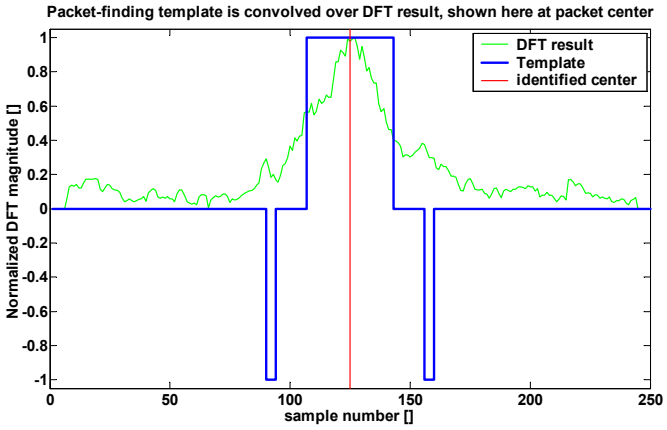

Figure 11: Non-rectangular packet-finding template

In summary, switching to such a non-rectangular template would produce a negligible increase in computation time (both are extremely efficient), provide slightly better ID accuracy for fringe packets away from the ends of the scan, but possibly significantly worse ID performance for fringe packets at or near the edge of the scan window.

Symmetry checking – The interferogram should ideally be symmetric about its center. The algorithm originally contained a step that would calculate the symmetry and weight the DFT (or envelope) result accordingly. Unlike virtually all other signal processing steps, the symmetry calculation could not be implemented with exceptional efficiency, and ended up taking about double the processing time of all other operations combined. Problems were also encountered with edge-effects; distortion would result when trying to calculate the symmetry when the fringe was partially off the scan window. The symmetry calculation was useful for gaining slightly more accuracy when the fringe was centered, but the combined issues of compute time and edge distortion led to its removal.

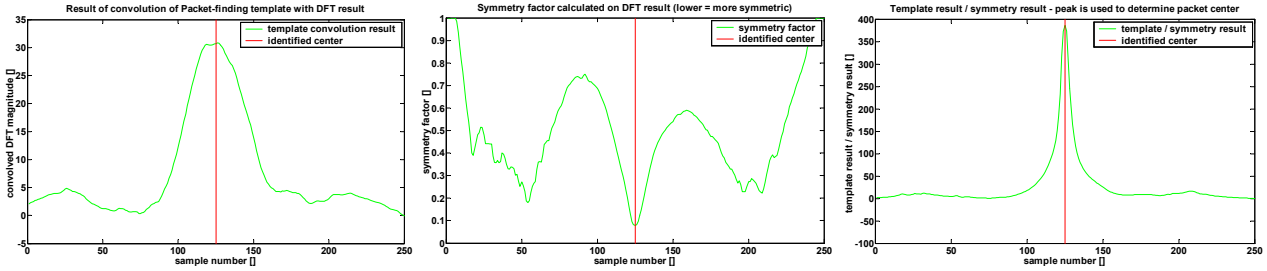

Figure 12: Symmetry checking to improve accuracy

The above sequence shows how the symmetry checking step was applied. The leftmost plot shows the output of the convolution of the packet-finding template with the composite DFT result. The middle plot shows the symmetry calculation (lower number means more symmetric). The rightmost plot combines the two, dividing the left result by the middle. The index corresponding to the maximum value of this result is used as the packet center estimate. This sequence illustrates the sharpening possible with this symmetry calculation.

Single frequency DFT – If the fringe frequency across the interferogram is consistent, then a single-frequency DFT can be used, rather than the combination of 5, as is presently used. Since the DFT calculation comprises the bulk of the computation, this would create close to a 5-fold decrease in compute time. However, it is unlikely that that would be a good tradeoff.

3. ENVELOPE-BASED TRACKING

The fringe-tracking algorithm developed in 1998-1999 and implemented in 1999-2000 used a different approach, which should still be considered as a viable option[6,7]. Developed and tested using about 4000 scans taken at IOTA in 1997, that algorithm was able to automatically identify fringe packet parameters to accuracy as good as could be determined by eye. That is, the identified fringe packet parameters (amplitude, spread of sinc function, center, frequency of fringes, and phase shift of fringes – or A, B, C, D, E from Equation 1) following envelope-based initialization and gradient based optimization appeared to be the "best" match to the actual data.

Step 1. Data sets from each of the two collectors are combined, for example using $y = \dfrac{channelA - channelB}{channelA + channelB}$.

Step 2. Outliers are removed

Step 3. Local bias is removed using a high-pass filter

Step 4. The envelope of the absolute value of the signal is calculated, eliminating the individual fringes. In the case of no noise, this envelope would have the form $y = A\,\mathrm{abs}(\mathrm{sinc}(B(x + C))$.

Step 5. A packet-finding template as described in Section 2.9 is convolved with this result.

Step 6. The initial estimate for the C is found by maximizing weighted symmetry over this result.

Gradient-based optimization, described Section 4, refines this C estimate, and identifies the other four interferogram parameters. A fringe tracker using steps 1, 3, 4, 5, and 6 was implemented at IOTA[7]. Morel converted the MATLAB code to C, then to Labview for implementation and integration with the other control system components.

4. GRADIENT-BASED OPTIMIZATION OF ALL FRINGE PARAMETERS

Following the use of either the DFT- or envelope-based algorithms to get an initial estimate of C, the remaining steps in identifying A, B, C, D, E use an FFT and nonlinear gradient-based optimization. These steps take about 200-600% longer to compute, and produce only a small improvement in fringe-packet-center identification. For the 4000-scan 1997 data set, the maximum difference between initial and final estimates for C over all the data was 1.6 samples; standard deviation was 0.5 samples.

The improvement in center-estimation alone probably does not warrant implementing the full estimation of A, B, C, D, E, since the random motion of the fringe packet center is roughly 20 times larger than this improvement. However, these additional parameters have been found to contain information useful for fringe-packet-motion prediction – initial tests show that past and present values of fringe frequency, D, are useful for predicting future values of interferogram center, C.

For consistency with the steps described in Section 3, step numbering begins where that ended.

Step 7. Using the initial estimate of C, and using an initial guess for B (sinc width), the remaining parameter describing the sinc envelope, A, is found by a least squares fit to the envelope found in step 4 of Section 3. This is a direct calculation.

Step 8. With A, B, C now having good initial estimates, an iterative gradient-based optimization is performed to find the A, B, C that minimize the squared error with respect to the envelope data.

Step 9. Fringe parameters, D and E, are found by fitting the ideal fringe function from Equation 1 to the data (following bias outlier and bias removal, Step 3 of Section 3) over the center of the fringe packet (half height of the sinc function determines the center region). An FFT taken over this packet center region provides initial guesses for D and E, and an iterative gradient-based optimization then finds C, D, E, with A and B held fixed.

Simultaneous gradient-based optimization of A, B, C, D, E was attempted, but did not work as well on the noisy data as the sequential procedure (A, B, C then C, D, E) listed above. One example of a cause of this difficulty is shown in Figure 13. Half-way through the fringe-packet scan, a sudden phase shift was encountered. If all 5 parameters were adapted simultaneously, the result would be a flat line, with $A = 0$, since the identification would be unable to lock onto the left and right halves of the fringe packet. Squared error would be minimized approximately by a flat line, $A = 0$. With the present algorithm, two parameters representing the sinc function envelope (A and B) were held fixed while the fringe frequency and phase shift (D and E) were identified. The identification locked onto the right half of the fringe packet since that resulted in a better fit than the left half. The figure shows the raw data, along with ID'ed sinc-function envelope and ID'ed full sinc-sinusoid function.

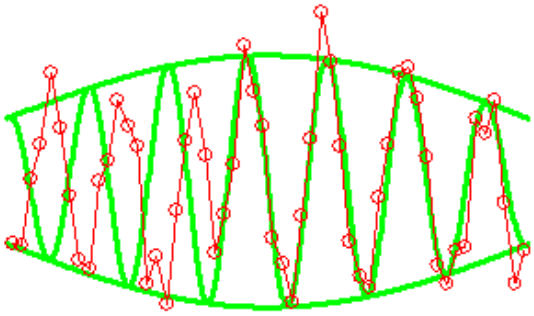

Figure 13: Example of significant intra-packet shift

It should be possible to calculate the gradient directly, rather than by perturbation (as is presently done). The perturbation calculation is not very inefficient here since there are only 3 parameters at a time being calculated (ABC or CDE). The main benefit from using a backpropagation-based gradient calculation is that computations do not increase linearly with the number of parameters. To do this, there would be a slight problem due to the absolute value function present in the calculation of A, B, C (since abs() is not continuously differentiable). This may be addressed by focusing only on the main packet, so the abs() could effectively be dropped from the equation.

5. PREDICTION OF FRINGE-MOTION

Initial, proof-of-concept experiments with fringe packet motion prediction were performed. An adaptive linear model using present and past identified fringe parameters was developed as a first step. The center of the next fringe packet and the magnitude of fringe-packet-center motion were predicted, with the goal of optimizing on-line the parameters (e.g., travel limits and rate) of the next scan. This model produced approximately 5% (and as high as 10%) improvement over no prediction (i.e., the baseline case of predicting that the fringe packet will remain where it is on the next scan). Extension to nonlinear prediction methods, including neural networks is under investigation.

In the data analysis that follows, different combinations of identified parameters were used, along with the changes in those parameters, to predict the change in interferogram center and the absolute value of the change. A least-squares linear predictor was used for each set of input parameters. To measure the prediction performance as expected on new data, vs. as fit to the training data, the results of each linear predictor were evaluated by testing on a segment of the data

that was not used during model development (test data vs. training data). This cross-validation procedure is common in model building and other machine learning applications.

The results of the prediction were compared against a baseline case that: (1) predicted no motion - that the interferogram would stay where it was, and (2) predicted that the absolute value of the motion would equal the RMS of all motion detected so far on that target. The percentage improvement as compared to these baselines is printed in the following table, which has each of 9 data sets from 1997 in rows (there are 8 ~500 point data sets, and the 9th set is all 8 combined), and each of the 8 input parameter combinations tested in columns.

	col1	col2	col3	col4	col5	col6	col7	col8	[Mean]
Data set 0	7.507	7.854	7.765	6.521	3.889	6.680	4.097	6.489	6.350
Data set 2	8.939	9.978	9.550	**10.232**	6.238	6.826	4.961	6.272	**7.874**
Data set 3	5.267	4.787	4.717	4.959	-0.818	5.280	4.673	2.829	3.962
Data set 5	2.312	2.786	3.883	3.583	3.556	3.841	2.804	2.739	3.188
Data set 6	7.506	7.581	7.453	7.502	3.026	4.301	6.893	2.850	5.889
Data set 7	0.778	-0.149	0.836	1.832	0.637	1.790	2.566	-0.279	1.001
Data set 8	-0.102	2.864	3.118	3.067	-0.350	3.017	3.861	2.768	2.280
Data set10	0.841	2.403	0.890	0.880	0.511	1.954	0.674	1.270	1.178
Data set11	2.940	2.675	3.081	3.074	1.476	2.953	2.931	1.259	2.548
[Mean]	3.998	4.531	4.588	**4.628**	2.018	4.071	3.718	2.911	

Variables used in prediction [y = A * sinc(B(x+C)) .* cos(D(x+E))]:

	A	B	C	D	E	A_delta	B_delta	C_delta	D_delta	ones_n_1
col 1 :			C	D			B_delta	C_delta	D_delta	ones_n_1
col 2 :			C	D			B_delta	C_delta		ones_n_1
col 3 :			C	D			B_delta	C_delta		ones_n_1
col 4 :			**C**	**D**				**C_delta**		**ones_n_1**
col 5 :				D				C_delta		ones_n_1
col 6 :			C					C_delta		ones_n_1
col 7 :			C	D						ones_n_1
col 8 :			C	D				C_delta		

6. IMPLEMENTATION AT IOTA

Since the tracker needs to run on a real-time processor (VxWorks operating system on a Motorola PowerPC 604 processor on a MVME-2431 card), after the initial development and prototyping in MATLAB, the algorithm was converted (manually) to ANSI C. As the algorithm evolved during this implementation process, the MATLAB and C versions were continually updated to maintain the same variable names, function names, and structure to the extent possible. The two versions produce results that are identical when compared to the limit of floating point precision (6 digits).

A test function was developed in both C and MATLAB that generated an idealized scan for testing the algorithm. This arrangement proved useful in determining whether a problem was in the interface between the calling function , the implementation of the C-version of the function, or the algorithm itself (in which case a problem would be detected when running in MATLAB). This development arrangement will enable off-line data analysis and facilitate further algorithm improvements.

The C code is compiled using WindRiver Tornado, which uses the gcc and g++ compilers to create a .out file that is then loaded onto the VxWorks target at run-time. Variables that may need to be changed, such as nFringe, nDft, and threshold_ratio, are passed as inputs to this function so that there should not be a need to recompile the fringe-tracking code if these parameters must be updated for different observing conditions. These parameters can be set through an IDL interface, enabling them to be changed easily during a run.

6.1. Testing

Initial testing was performed during February 2002 on the IOTA system, tracking fringes generated by a light source. Tracking performance was very good, even with temporary loss of fringe data (for example, caused by banging the table) – in these cases, the system correctly decided that confidence was low and did not try to track until the fringe packet re-formed. Also, the system performed very well with the scan travel set at both 15 and 30 microns, and with no manual adjustment of parameters. The fringe packet appears twice as wide for the 15-micron scan, further indicating the robustness of the algorithm. Unfortunately, due to poor weather conditions, we were unable to test it on the sky.

When initially implemented, some overshoot was noticed when the fringe tracking was active. This was effectively addressed by implementing a gain of 0.7 in the control loop. So instead of commanding a motion to move to the identified center location, the system moves 70% of that distance.

6.2. Speed

The following results were obtained by running the function for 10,000 or 100,000 times and measuring the total elapsed time, and allowing for a calibrated conversion between the PC used for testing and the real-time processor. The algorithm generated an ideal fringe packet (time to compute this is included) and then identified the center. Running the full algorithm presented in Section 2 took 0.67 milliseconds per cycle on the PowerPC. Including all three interferograms for each scan, the total compute time is 2.0 milliseconds, which is fast enough to be used.

% of time	time [ms]	Algorithm step(s)
75%	0.50	4 extra DFT calculations for window size adaptation
19%	0.13	Required DFT calculation
6%	0.04	Everything else (template, confidence, etc.)
100%	0.67	Total time per interferogram per scan

7. CONCLUSIONS

Interferogram center ID works very well on the data sets tested so far, including the 1997 data, data generated by a light source using the IOTA configuration as of Feb 2002, and actual on-the-sky fringes from 2002 and 2004. The fringe-center estimate provided by the Adaptive DFT-based or envelope-based algorithms is sufficiently accurate for fringe tracking. That is, the initial estimate of C in the equation $y = A\ \mathrm{sinc}(B(x+C))\ \cos(D(x+E))$ is sufficient, and full nonlinear estimation of A, B, C, D, E provides only marginal improvement. The adaptive nature of the DFT-based algorithm virtually eliminates the need to set any target-dependent parameters, and provides robust tracking in the presence of significant atmospheric distortion.

On-line implementation of this algorithm at IOTA was completed in May 2002, using all three telescopes. Compute time for all three interferograms is estimated to be 2.0 ms. Fringe tracking was considered successful, but no quantitative results have yet been derived from these tests.

Prediction of interferogram motion using adaptive linear predictors appears feasible to a small extent, but major effort in this area may not be warranted since the payoff appears small relative to that achievable through fringe tracking control (without prediction).

ACKNOWLEDGEMENTS

The algorithm development work presented here was funded through Director's Discretionary Fund awards at NASA Ames Research Center. The authors wish to thank the staff and other researchers at IOTA for their invaluable contributions to the research facility. The IOTA is operated by the Smithsonian Astrophysical Observatory, a member of the Harvard-Smithsonian Center for Astrophysics.

REFERENCES

1 W.A. Traub, et al, "The third telescope project at the IOTA interferometer," *Proc. SPIE 4006*, 2000.
2 W.A. Traub, et al, "New beam-combination techniques at IOTA," *Proc. SPIE 4838*, 2002.
3 http://cfa-www.harvard.edu/cfa/oir/IOTA/, 2004
4 R. Millan-Gabet, *Investigation of Herbig Ae/Be stars in the near-infrared with a long baseline interferometer*, Ph.D. thesis, University of Massachusetts at Amherst, 1999.
5 E. Pedretti, *Systèms d'Imagerie Interférométriques (Imaging Interferometric Systems)*, Ph.D. thesis, Université de Provence - Aix-Marseille I, Observatoire de Haute-Provence, France, 2003.
6 E. Wilson and R. Mah, "On-line fringe tracking and prediction at IOTA," in *Proceedings of the 18th Congress of the International Commission for Optics*, San Francisco, California, August 1999.
7 S Morel, et al, "Fringe-tracking experiments at the IOTA interferometer," *Proc. SPIE 4006*, 2000.
8 E. Pedretti, et al, "Fringe tracking at the IOTA interferometer," *Proc. SPIE 5491-62*, 2004.
9 N. Thureau, et al, "Fringe envelope tracking at COAST," *Proc. SPIE 4838*, 2003.
10 E. Wilson, et al, "Adaptive DFT-based interferometer fringe tracking," in *EURASIP Journal on Applied Signal Processing – special issue on Applications of Signal Processing in Astrophysics and Cosmology*, Q2, 2005 (submitted).