Submitted to *EURASIP Journal on Applied Signal Processing*,
special issue on Applications of Signal Processing in Astrophysics and Cosmology

Adaptive DFT-based interferometer fringe tracking

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

submitted: June 1, 2004

ABSTRACT

An automatic interferometer fringe tracking system has been developed, implemented, and tested at the Infrared Optical Telescope Array (IOTA) observatory at Mt. Hopkins, Arizona. The system can minimize the optical path differences (OPDs) for all three baselines of the Michelson stellar interferometer 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. The adaptive DFT-based tracking algorithm should be applicable to other systems where there is a need to detect or track a signal with an approximately constant-frequency carrier pulse.

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 in [Traub 2000] [Traub 2002] [IOTA 2004].

This article reports on the development of an algorithm designed and used to simultaneously null the optical path differences (OPDs) for the three baselines (A-B, A-C, and B-C) provided by IOTA's three apertures.

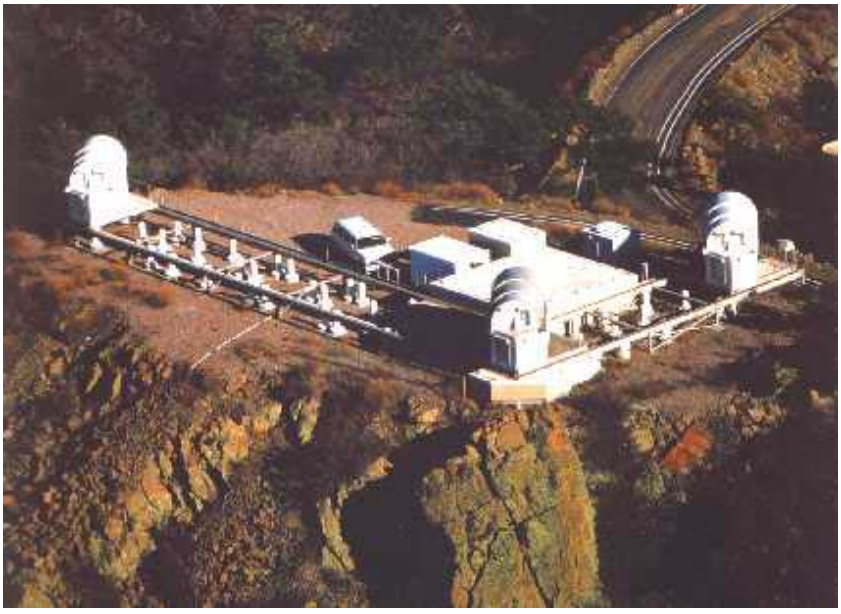


Figure 1: Infrared-Optical Telescope Array (IOTA)

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

1.1. Fringe tracking goals

Details of the relevant interferometric derivations is covered thoroughly in other references such as [Millan-Gabet 1999] and [Pedretti 2003]. From a signal processing perspective, it is important to know that the governing physics of stellar pupil-plane Interferometry result in an ideal signal looks like that shown in Figure 2.

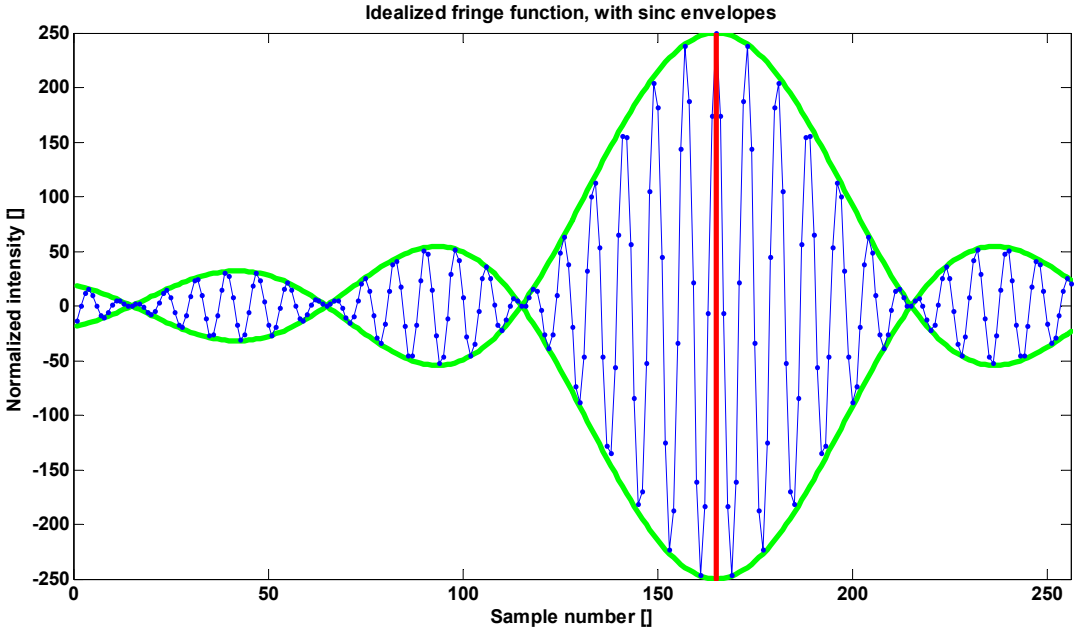


Figure 2: Idealized interferogram, sinc-function envelopes and center shown

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 gradient-based optimization procedure described in [Wilson 1999] and [Wilson 2004]. Basically, the sinusoid in this function comes from the interferometric combination of two sources (a telescope pair), and the sinc function enters due to the Fourier transform of the instrument's spectral response (which is uniform over a fixed range). Although E is not an independent variable in theory, in practice, noise and atmospheric disturbances make it independent of the other parameters.

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

The center of the fringe packet, or interferogram, corresponds to the point at which the optical path difference (OPD) between each of the two collectors and the target (star) is zero. The path lengths are adjusted with slow- and fast-moving mirrors, driven to account for slow (and well calculable) effects such as earth rotation, as well as fast effects such as atmospheric turbulence. The interferogram represents the samples taken as a piezo-driven mirror is driven through a stroke of typical length 30 microns over a period of typically 300 ms. If the center of this mirror scan stroke is not sufficiently close to the true OPD zero point, the fringe packet will be lost from view, and no science data will be available.

So the goal of an interferometer fringe tracker is to analyze incoming interferograms, and provide on-line adjustments to the piezo-scanning delay-line mirror to keep the fringe packet centered within the scan window. It should be:

1. Robust to noise and anomalies in the data – absolute accuracy is not as important as keeping the fringe within the scan window

2. Require few if any manual adjustments – autonomous adaptability is needed to cover widely varying seeing conditions and object intensities
3. Computationally efficient – requiring a minimal amount of computation time due to the limited resources and need for fast scanning, typically 3 Hz.

Maximum accuracy is less important than robustness, since as long as the fringe packet is in the scan window, it can be analyzed in post-processing. As fringe tracking accuracy increases, however, it becomes possible to reduce the stroke length of the piezo scanner, thus increasing the overall rate of data collection. There is a secondary benefit in reducing stroke length in that, for a constant scan velocity, a shorter stroke will mean less time between scans, which reduces the time during which the atmosphere may have changed, reducing the average size of the fringe packet random motion.

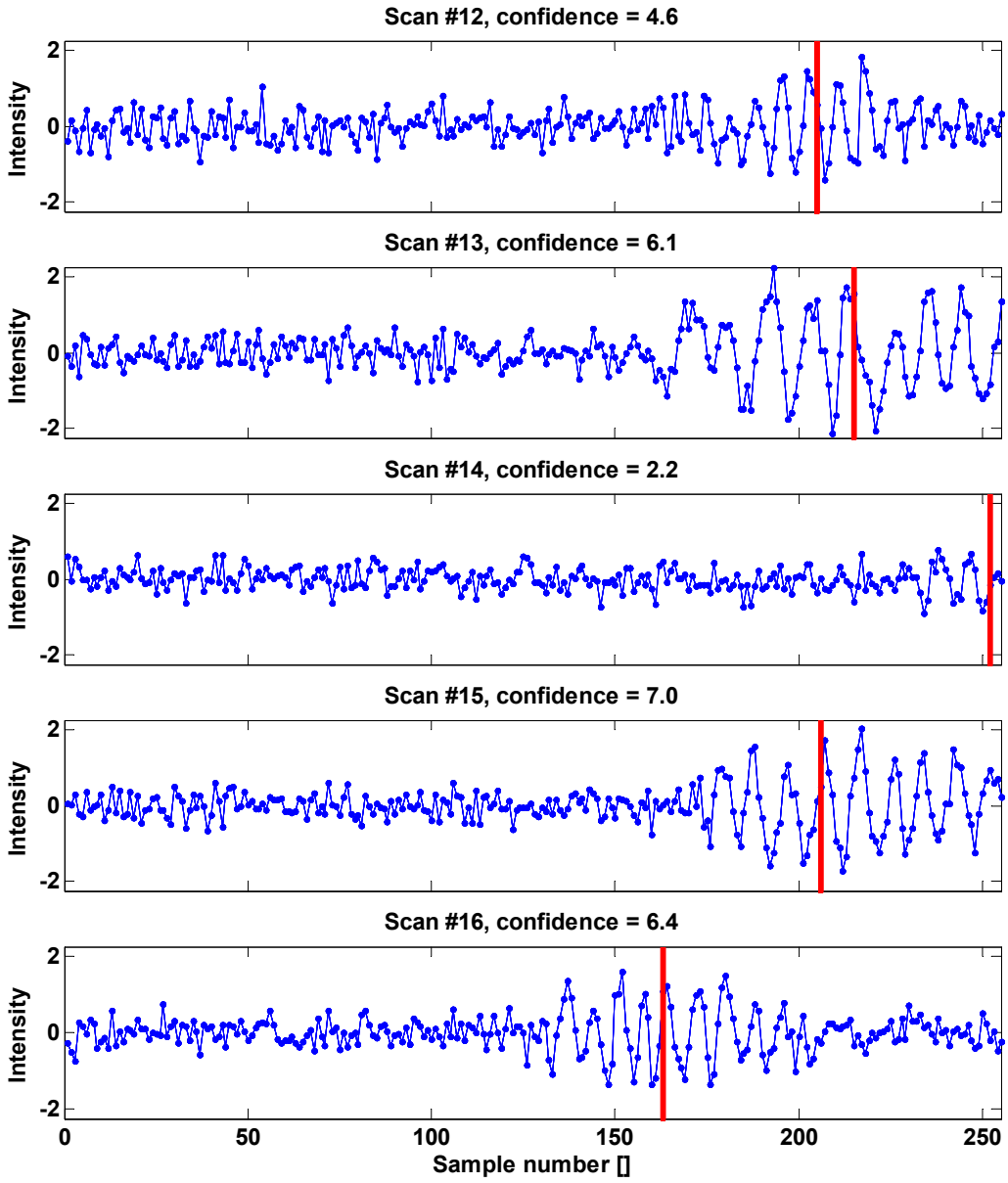


Figure 3: Typical sequence of scans

Figure 3 shows a typical sequence of scans from IOTA. This data was taken on April 20, 2004, on the B-C telescope pair, targeting star HD126035, with the fringe tracker turned off. These are 5 consecutive scans out of the 200-scan

data set, taken at a scan rate of 3 Hz, with a 25 micron scan length. Although the idealized form of the sinc-sinusoid from Equation 1 can be seen, there is significant background noise, variability of the sinusoid (fringe) frequency from scan to scan, and within each scan, and the packet center can be seen to move randomly between one scan and the next. Although these interferograms appear to have relatively consistent quality (with the exception of scan #14, which has drifted almost completely out of the window), it is not uncommon to have significant changes in quality (noise level, jump size, fringe clarity) from one scan to the next.

Along with the raw data shown in the figure, the solid vertical line indicates the identified fringe packet center, that could have been used by the fringe tracking software to re-center the piezo scan, had fringe tracking been turned on. The relative confidence in the fringe center identification is indicated the title of each sub-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. The computing and actuation aspects of the control system are described by Traub and Pedretti [Traub 2002] [Pedretti 2003]; the present article details the fringe tracking algorithm and aspects of its software implementation.

Due to the noise sources present and the lack of a sufficiently representative simulation, 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).

1.2. 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 [Wilson 1999]. 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 [Morel 2000]. 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 [Traub 2002].

Pedretti developed a fringe tracking algorithm taking a completely different approach, based on double Fourier Interferometry (DFI) [Pedretti 2003,2004]. 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 [Thureau 2003].

Gradient-based optimization, motion prediction, and other off-line analyses are discussed in [Wilson 2004]. As compared to that publication, the present article uses data from 2004 and focuses on the adaptive DFT-based tracking algorithm, whereas [Wilson 2004] focuses on the IOTA implementation issues, envelope-based tracking, and off-line gradient-based optimization of all packet parameters.

1.3. 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 [Morel 2000].

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). See, for example, scan #14 in Figure 3: in that case, the envelope would not be a clear signal, but focusing on the expected fringe frequency leads to an accurate identification even with very little of the fringe packet in the window. 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.

2. DFT-BASED TRACKING

The algorithm is summarized in Section 2.1, then the individual steps are outlined in subsequent sections.

2.1. DFT-based tracking 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. Example data

The algorithm steps are presented using actual data, as shown in the following figures. They were generated using data collected from the BC-fringe of the 16th scan of the IOTA-25 dataset on April 20, 2004; targeting star HD126035; RA (J2000): 14.390278; Dec (J2000): -11.713889. This interferogram is also shown in Figure 3.

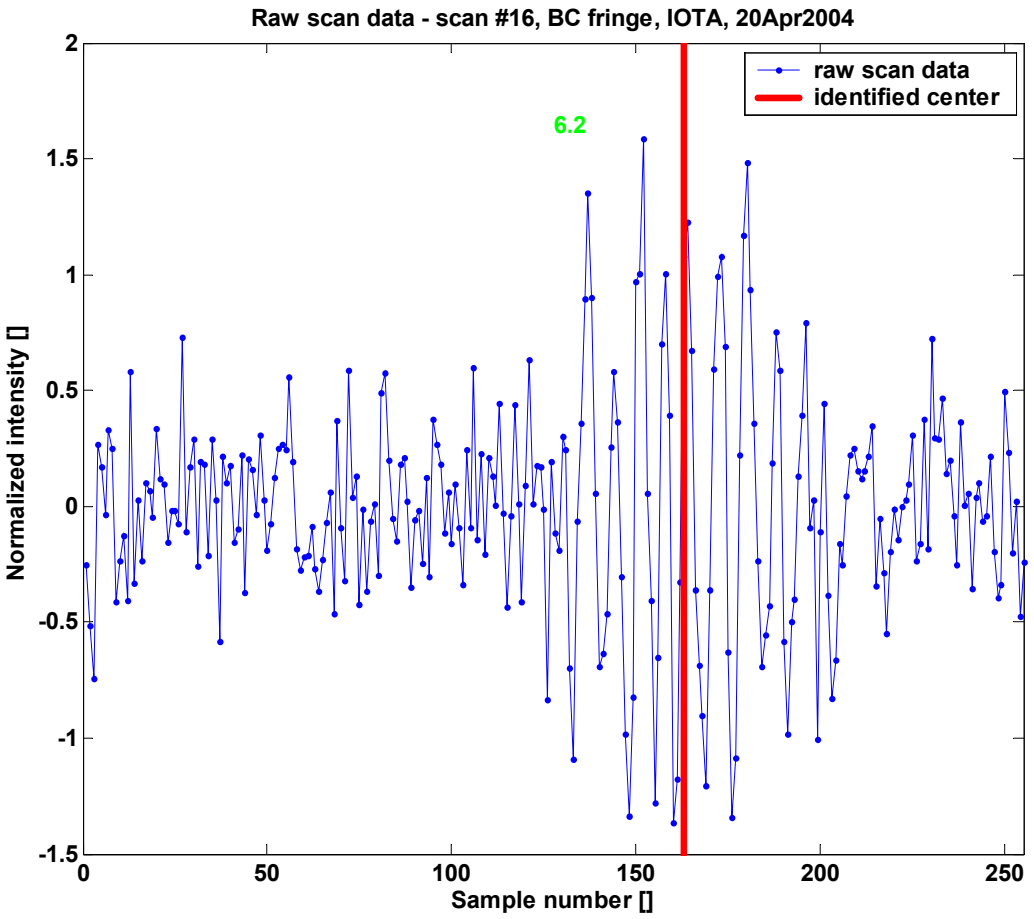


Figure 4: Normalized raw data

Figure 4 shows the normalized raw data (the complementary pair, B-C), as well as the result of the center identification that came after all steps were completed. The DFT-based confidence metric (6.2 in this case) is shown as well as text.

2.3. DFT calculation

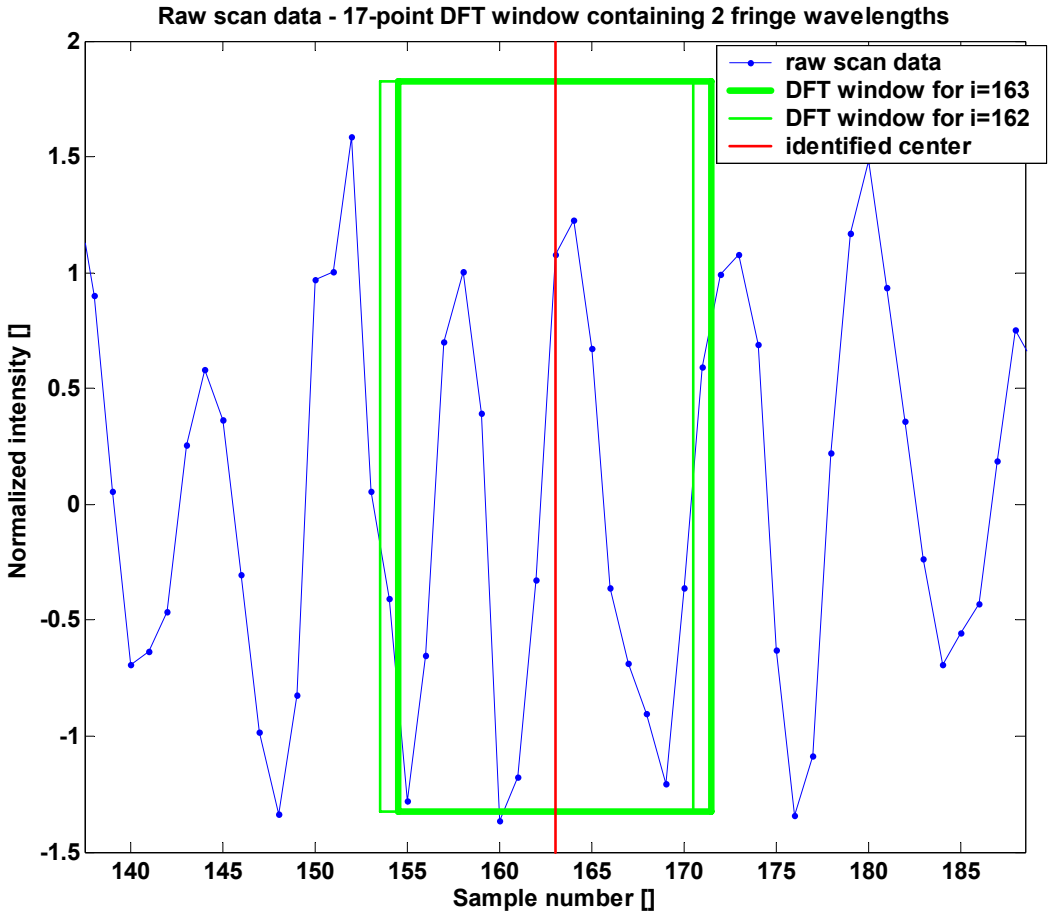


Figure 5: DFT sliding window

Figure 5 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 #163 in the figure uses the DFT result for sample #162, then adds a term for point #171 (= 163 + (17-1)/2, the new point in the sliding window) and subtracts a term for point #154 (= 162 - (17-1)/2, the point dropping out of the sliding window).

While it is efficient, the DFT calculation is also exact. By doing this calculation only for only the frequency component of interest, this is faster than an FFT. FFT compute time is proportional to $N \log_2 N$, where N is the window length, a full-spectrum DFT would be N^2, while this single frequency DFT compute time is proportional to N. But a second level of optimization is achieved by using a rectangular window and sliding it across the scan, one sample at a time, as follows:

- For implementation in C, the real and imaginary parts are calculated separately. So sine and cosine functions corresponding to the fringe frequency (as defined by the window size and bin number) are multiplied by the entire data scan (e.g., 256 points) - once per scan.
- The real and imaginary parts of the DFT result for the first sample in the scan is calculated by summing the above-calculated real and imaginary vectors over a window.
- Then, to calculate the real and imaginary DFT results at successive samples throughout the middle of the scan, the calculation is just an add and a subtract for the real and imaginary parts. In particular, this is possible because a rectangular window is used.
- Since only the magnitude (vs. the phase) is used, the real and imaginary results are combined accordingly as the final step.
- All computations are done using floats (vs. doubles), since this provides sufficient accuracy.
- The DFT calculation does not depend at all on whether the window or scan size is a power of 2 (as the FFT does to some extent – although the non-power-of-two inefficiency is very slight for some FFT implementations such as FFTW).

The DFT window size is chosen in this case to nominally contain exactly two fringe wavelengths. The algorithm requires the window size to be an odd integer whose selection is discussed later. Because of the way the DFT-calculation algorithm has been implemented, window sizes of 3, 4, 5, etc. wavelengths could be calculated without changing the computation time. However, for larger windows, there is a possibility of the frequency changing during the window, which would distort the DFT calculation (i.e., when the time corresponding to the window size is less than the coherence time). 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 6. This step is computationally efficient, and reduces the variability in the DFT results.

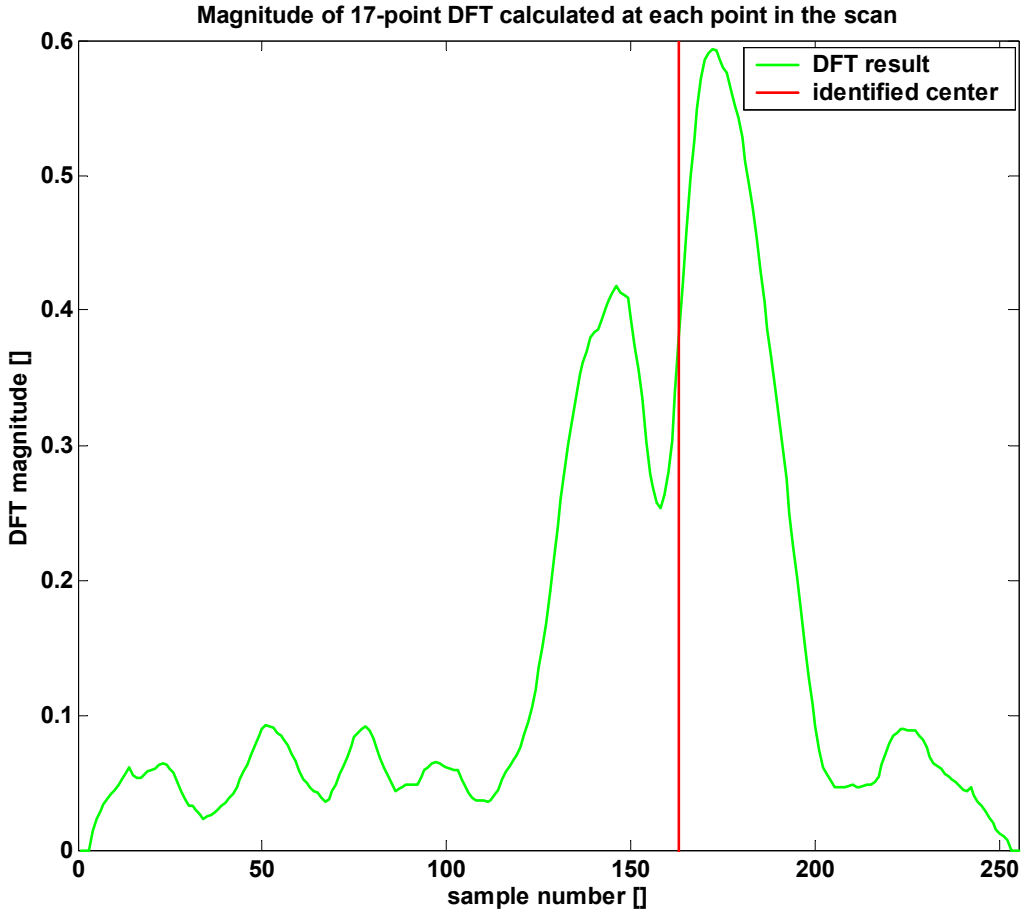


Figure 6: Smoothed DFT result, using, in this case, a 17-point sliding window

Even though it is calculated very differently, the result is very similar to that resulting from the envelope-finding calculations described in [Wilson 1999]. While the envelope-finding calculations provided excellent results, this DFT calculation is more physically based and is expected to provide better robustness for noisy signals.

The DFT can be thought of as a sampled version (exists only at the bin frequencies) of the discrete-time Fourier transform (DTFT) of the continuous signal convolved with the DTFT of the window function (in the simplest case, as we have here, this is a uniform window of finite length). With an infinitely wide window, the DTFT of the window would be an impulse, so the convolution would not distort the DTFT of the signal. With a finite window, two effects occur:

1. **Reduced resolution** - unlike an impulse, the main lobe of the window function has some finite width. Convolving this with the signal DTFT may make it impossible to resolve between two frequency components.
2. **Leakage** - the component at one frequency leaks into that at another component due to the spectral smearing.

If the goal is to measure the DTFT of the signal, then ideally one would like a window with a DTFT of a thin main lobe and small side lobes, but usually there is a trade off between the two. A rectangular window has a relatively narrow main lobe, while Hanning, etc. windows have a wider main lobe (worse), but have smaller side lobes (better). No matter the shape of the window, the main lobe gets narrower as the number of points in the DFT increases. For non-stationary signals as we have here, at some point you don't want to increase the window size

because the frequency content is changing. In this case, the length and shape of the window are chosen so that the Fourier transform of the window is narrow in frequency compared with changes in the FT of the signal [Oppenheim 1989].

However, in this application, the main concerns are not with resolution or leakage since our target, the fringe frequency, is changing from scan to scan and within a scan. Reduced resolution actually helps insulate the result from these variations. By adapting to these fringe frequency changes, the DFT result is useful for fringe tracking.

2.4. Combination of DFT results at multiple frequencies

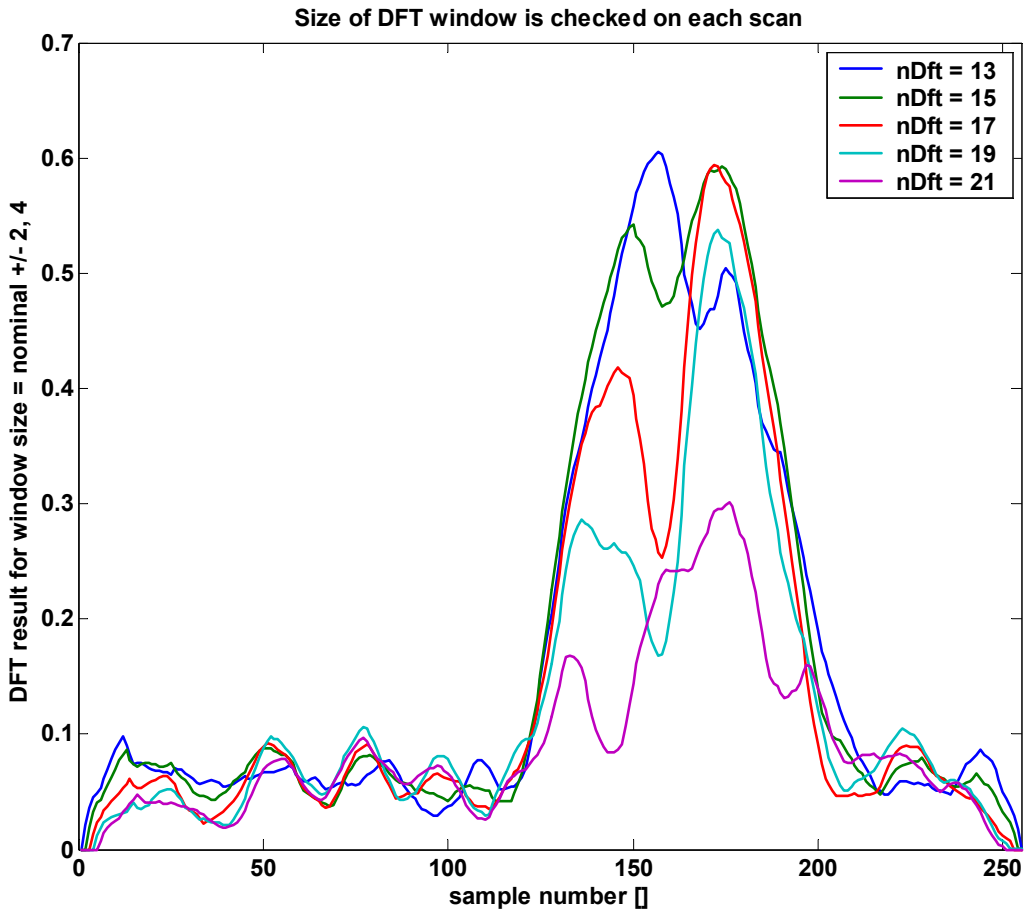


Figure 7: Smoothed DFT results for 13, 15, 17, 19, and 21-point windows

Figure 7 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 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. Also, to prevent unnecessary changes, the nominal nDft is changed only if the maximum exceeds that of the nominal by 5%. So in the example shown, where the highest DFT result occurs for nDft = 13 at sample #157, it was within that threshold, so the nominal nDft carried forward was 17.

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 7) 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.

2.5. Convolution with packet-finding template

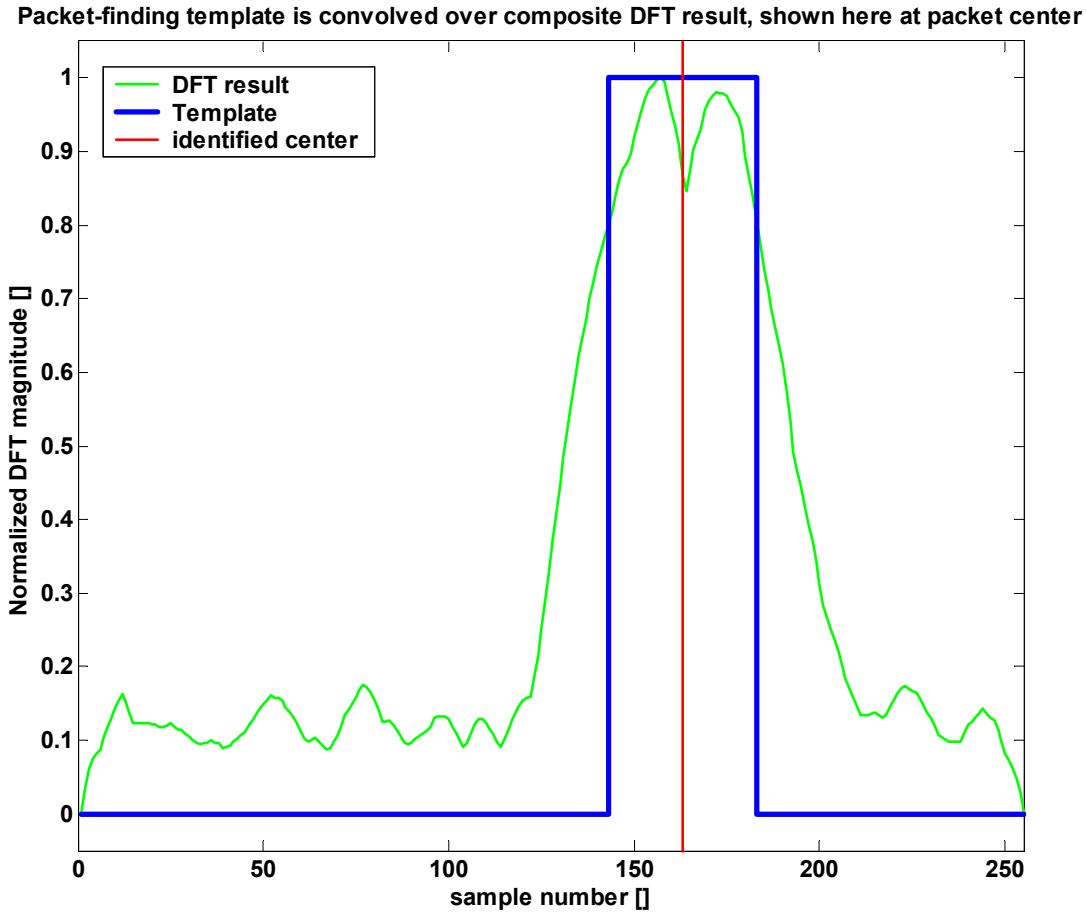


Figure 8: Composite DFT result prior to convolution with a rectangular packet-finding window

Figure 8 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 20 for the half-width (meaning the template spans 41 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).

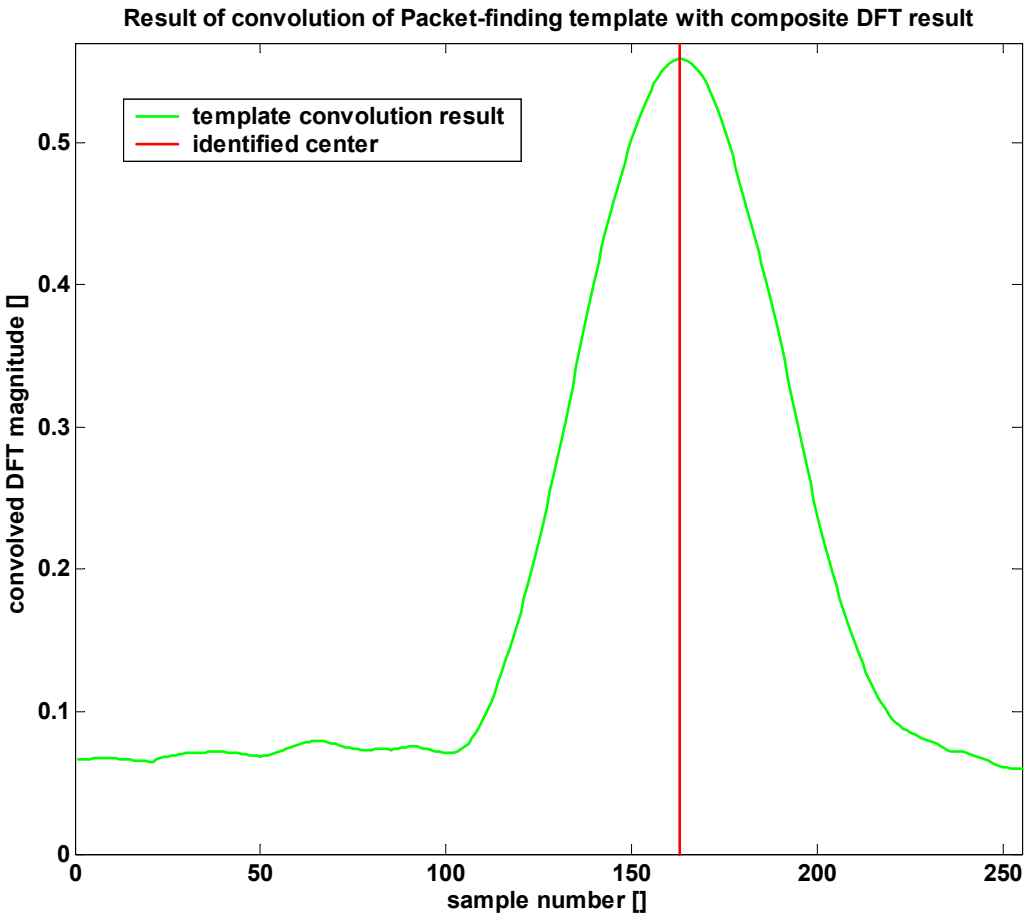


Figure 9: Composite DFT result after convolution with a rectangular packet-finding window

Figure 9 shows the result of convolution with the rectangular template. The index corresponding to the maximum value of this result is used as the packet center estimate.

2.6. Confidence metric calculation

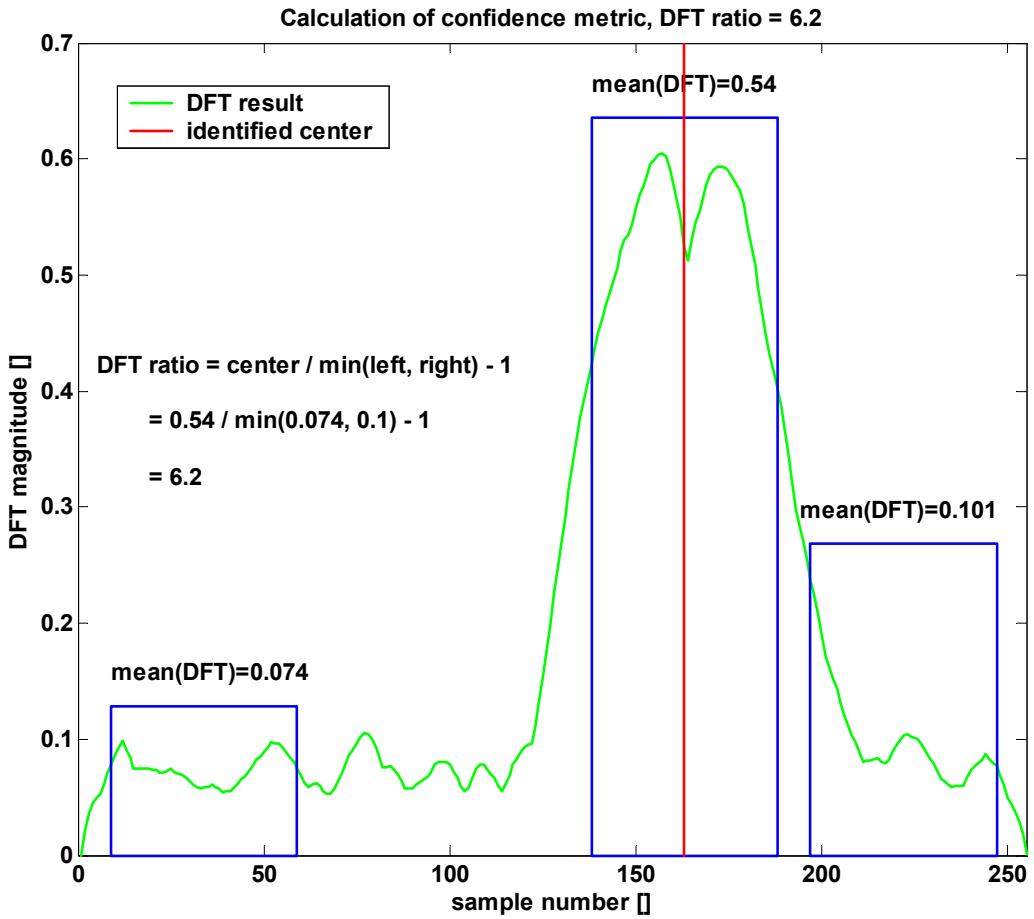


Figure 10: Confidence metric calculation

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.

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.

2.7. Results on scan with lower SNR

While the preceding figures have illustrated the algorithm's functional steps, it is also useful to see how the algorithm performs on data with a lower SNR, as is shown in Figure 11 (this scan is also shown in Figure 3). To reduce clutter, some titles and axis labeling is omitted – see the preceding figures for clarification. The DFT analysis clearly and effectively detects the fringe frequency in this noisy signal. The confidence metric calculation provides a meaningful comparison between the detected fringe packet and the background noise.

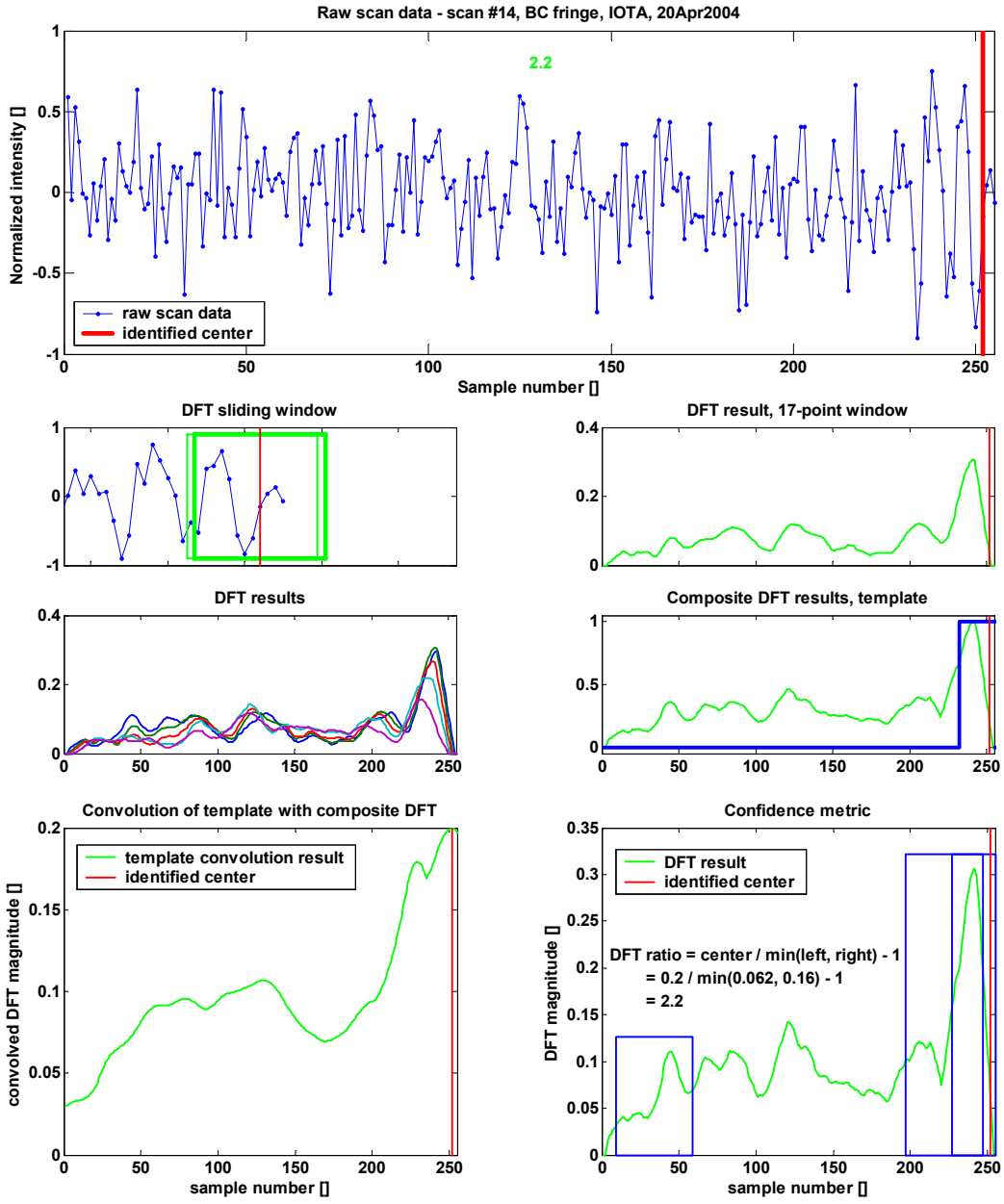


Figure 11: Result for scan with lower SNR

2.8. 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 12. 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.

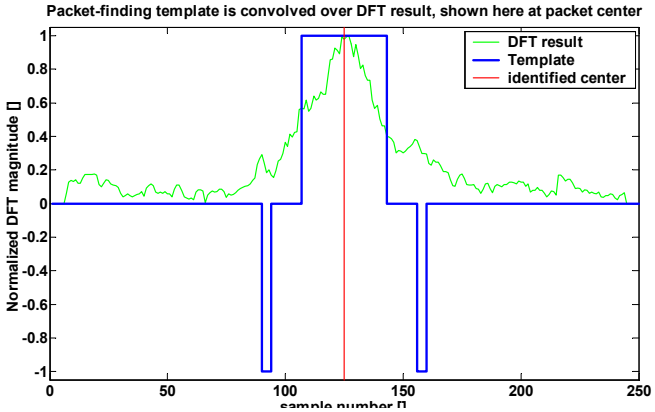
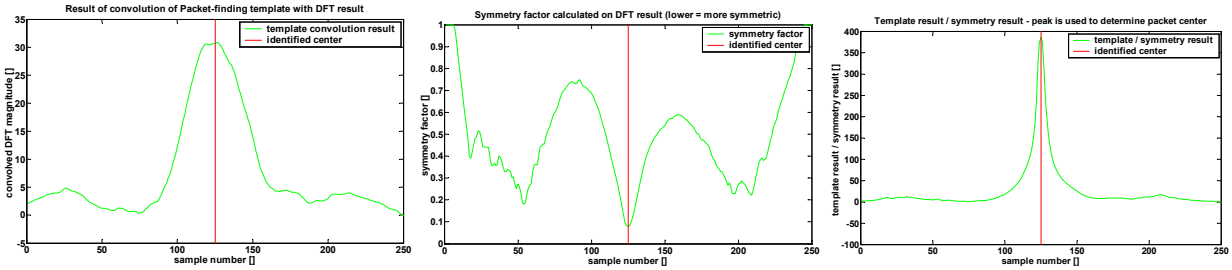


Figure 12: Non-rectangular template

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.



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.

2.9. Potential algorithm improvements

There are some areas where known improvements in accuracy could be made at the cost of code complexity and development effort.

- **A priori expected position** - using the expected fringe motion, and knowing whatever delay line motion has occurred, there will be an expected fringe center before the scan is analyzed. For example, if zero motion is assumed, and the control drives exactly to the previous center, then the expected fringe center would be in the middle of the scan. Some benefit should be gained from factoring this information in to the tracking algorithm, perhaps by multiplying a weighting function (perhaps exponentially decaying away from the center, with the decay rate based on the observed volatility of the motion and the confidence in the

prior estimate) by the composite DFT result. This would hopefully generally have little additional effect, but when faced with a choice between two weak peaks, would choose the one closer to the expected location.

- **Prediction** - Add prediction to the a priori estimate above, as well as to the ptracker output - i.e., right now we output the identified position of the current scan and the associated confidence. What is really wanted is the expected position of the next interferogram and the confidence of that expectation. It seems the improvement may be slight, but it would be pretty easy to do, and the linear proof of concept shows measurable predictability of the fringe motion, even with a 3 Hz scan frequency (would be more predictable with faster scanning). The linear tests also indicate that the jump variance may be somewhat predictable – this could be incorporated into the exponential weighting function.
- **Check jump size** – A simple check could be added to detect a large jump with low confidence – in which case, the search space could be narrowed to a smaller region about the expected center. This is a simpler version of the previous two algorithm improvements, and may not be relevant if those are implemented.
- **Edge effects** - these are not handled very accurately in the present algorithm, since it is considered that they are less important (i.e., if we're on the edge, it is important to take a good jump towards the middle, but hitting it exactly is not a real goal). If performance near the edges is important, the algorithm can be implemented more carefully in these areas.
- **Adapt nominal interferogram width** – the expected width of a fringe packet is presently a fixed quantity, and the algorithm is fairly robust to adjustments to this setting. However, if adapted, the algorithm could be even more robust. If the full ABCDE identification is performed, the B (spread of sinc function) could be used to adaptively update this.
- **Startup** – Since the algorithm is adaptive (the size of the DFT window is adapted from scan to scan) and the amount of adaptation permitted in a single scan is purposely limited, the performance on the first couple of scans in a new data set is sometimes not as good as it could be. It is possible to allow some extra adaptation steps on the first scan of a new data set, if this is a real issue.

3. 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.

3.1. 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.

3.2. 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

4. CONCLUSIONS

An algorithm to perform on-line interferogram center identification has been developed, implemented, and tested at IOTA. It works very well on the data sets tested so far, including 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 adaptive nature of the DFT-based algorithm virtually eliminates the need to set any target-dependent parameters, and provides robust, accurate 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. The efficient algorithm design resulted in a compute time for all three interferograms of 2.0 ms, when implemented in ANSI C on the PowerPC 266 MHz real-time processor. Fringe tracking was considered successful, but no quantitative results have yet been derived from these tests.

REFERENCES

[IOTA 2004] http://cfa-www.harvard.edu/cfa/oir/IOTA/, 2004

[Millan-Gabet 1999] 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.

[Morel 2000] S Morel, et al. "Fringe-tracking experiments at the IOTA interferometer," *Proc. SPIE 4006*, 2000.

[Oppenheim 1989] A. Oppenheim and R. Schafer, *Discrete-time signal processing*, Prentice Hall, 1989.

[Pedretti 2003] E. Pedretti, *Systèms d'Imagerie Interférométriques (Imaging Interferometric Systems)*, Ph.D. thesis, Université de Provence - Aix-Marseille I, Observatoire de Haute-Provence, France, 2003.

[Pedretti 2004] E. Pedretti, et al, "Fringe tracking at the IOTA interferometer," in *Proceedings of the SPIE Astronomical Telescopes and Instrumentation Symposium: New Frontiers in Stellar Interferometry*, June, 2004, Glasgow, Scotland (SPIE vol. 5491-62).

[Thureau 2003] N. Thureau, et al, "Fringe envelope tracking at COAST," *Proc. SPIE 4838*, 2003.

[Traub 2000] W.A. Traub, et al, "The third telescope project at the IOTA interferometer," *Proc. SPIE 4006*, 2000.

[Traub 2002] W.A. Traub, et al, "New beam-combination techniques at IOTA," *Proc. SPIE 4838*, 2002.

[Wilson 1999] 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.

[Wilson 2004] E. Wilson, et al, "Adaptive DFT-based fringe tracking and prediction at IOTA," in *Proceedings of the SPIE Astronomical Telescopes and Instrumentation Symposium: New Frontiers in Stellar Interferometry*, June, 2004, Glasgow, Scotland (SPIE vol. 5491-173).