Published in final edited form as:

IEEE Trans Ultrason Ferroelectr Freq Control. 2008 September ; 55(9): 2084–2091. doi:10.1109/TUFFC.900.

Computationally Efficient Spline-Based Time Delay Estimation

Francesco Viola and William F. Walker

Department of Biomedical Engineering, University of Virginia, Charlottesville, VA 22908

Abstract

We have previously presented a highly accurate, spline-based time delay estimator (TDE) that directly determines sub-sample time delay estimates from sampled data. The algorithm uses cubic splines to produce a continuous time representation of a reference signal, then computes an analytical matching function between this reference and a delayed signal. The location of the minima of this function yields estimates of the time delay. In this paper we present more computationally efficient formulations of this algorithm. We present the results of computer simulations and ultrasound experiments which indicate that the bias and the standard deviation of the proposed algorithms are comparable to those of the original method, and thus superior to other published algorithms.

I. INTRODUCTION

Time Delay Estimation (TDE) lies at the heart of signal processing algorithms in a broad range of application areas including communications, coherent imaging, speech processing, and acoustics. Because of its central significance, TDE has been widely and meticulously studied over the past forty years [1,2]. Consequently, a variety of algorithms have been developed, each balancing accuracy, precision, and computational cost [3,4,5].

We have previously described a time delay estimator that directly determines continuous time delay estimates from equally spaced sampled data [6]. To summarize, given equally spaced sampled reference and delayed signals $s_1[i]$ and $s_2[i]$ with lengths N and M respectively ($N > M$), the technique forms a spline-based piecewise continuous representation of the reference signal using cubic splines. This representation is given by:

$$s_1(t) = \begin{cases} f_1(t) = a_1(t/\partial - T_1)^3 + b_1(t/\partial - T_1)^2 + c_1(t/\partial - T_1) + d & 0 \leq t < \partial \\ \vdots & \vdots \\ f_i(t) = a_i(t/\partial - T_i)^3 + b_i(t/\partial - T_i)^2 + c_i(t/\partial - T_i) + d_i & (i-1) \cdot \partial \leq t < i \cdot \partial \\ \vdots & \vdots \\ f_{N-1}(t) = a_{N-1}(t/\partial - T_{N-1})^3 + b_{N-1}(t/\partial - T_{N-1})^2 + c_{N-1}(t/\partial - T_{N-1}) + d_{N-1} & (N-2) \cdot \partial \leq t < (N-1) \cdot \partial \end{cases} \quad (1)$$

where ∂ is the sampling interval, a_i , b_i , c_i , and d_i are the cubic spline coefficients, and $T_i = (i - 1)$. In order to estimate the time delay between the two signals (one sampled, and the other represented analytically using cubic splines) we compute their sum squared error (SSE), as given by the following equation:

$$\varepsilon(t) = \sum_{i=1}^M (f_i(t) - s_2[i])^2 \quad (2)$$

The value of t that minimizes the SSE, $\varepsilon(t)$, is the local delay estimate. This estimate is found by finding the roots of the derivative of $\varepsilon(t)$ with respect to t . This estimation process is repeated for every overlap of the delayed signal (i.e., the kernel) within the reference signal (i.e., the search), yielding a vector of local time delay estimates. The global time delay estimate, representing the shift in time between the reference and the delayed signals, is the element from the vector of local estimates that has a value lying between 0 and 1, indicating that the true delay was within the tested interval. Results using simulated ultrasonic radio frequency (RF) signals clearly show that the proposed algorithm significantly outperforms other well-established algorithms in terms of jitter and bias over a broad range of conditions [6].

In this paper we develop more computationally efficient versions of the algorithm. Computer simulations and experimental data are used to evaluate performance.

II. PROPOSED METHODS

The algorithm described above, while highly accurate and precise, entails a substantial computational cost. The most costly steps are the generation of the spline coefficients and the root finding of the 5th order polynomial (derivative of the error function). Here we propose more rapid approaches to each of these problems.

1. Generation of spline coefficients

Given a set of data points $s_1[i]$, spline coefficients can be determined by solving a set of simultaneous linear equations that enforce equality between the splines and the data points, first and second derivative continuity at the points where piecewise polynomials meet, and some arbitrary boundary conditions (in this paper we use the not-a-knot end condition) [7]. This method yields a symmetric tri-diagonal system of equations that can be solved to generate the coefficients of equation (1)[7]. Solution of this system of equations entails multiplication of the N data samples with a sparse matrix of dimensions $4(N-1) \times N$. In this paper we explore the use of more efficient algorithms to generate the required coefficients.

The first method uses finite impulse response (FIR) filters to derive the coefficients of equation (1)[8, 9, 10]. These filters can be obtained by inverting the tri-diagonal matrix which imposes the continuity conditions described above. The weights of the filters can be determined from the rows of the inverted matrix and are given by the following set of equations:

$$\begin{aligned} \text{fir1}[k] &= \begin{cases} (15 - 9 \cdot \sqrt{3}) \cdot (-2 + \sqrt{3})^{-1-k} & k < 0 \\ -4 + 3 \cdot \sqrt{3} & k = 0 \\ 4 - 3 \cdot \sqrt{3} & k = 1 \\ -(15 - 9 \cdot \sqrt{3}) \cdot (-2 - \sqrt{3})^{2-k} & k > 1 \end{cases} \\ \text{fir2}[k] &= \begin{cases} (-9 + 6 \cdot \sqrt{3}) \cdot (-2 + \sqrt{3})^{-1-k} & k < 0 \\ 3 - 3 \cdot \sqrt{3} & k = 0 \\ (-9 + 6 \cdot \sqrt{3}) \cdot (-2 + \sqrt{3})^{1-k} & k > 0 \end{cases} \\ \text{fir3}[k] &= \begin{cases} -(6 - 3 \cdot \sqrt{3}) \cdot (-2 + \sqrt{3})^{-1-k} & k < 0 \\ 0 & k = 0 \\ (6 - 3 \cdot \sqrt{3}) \cdot (-2 - \sqrt{3})^{1-k} & k > 0 \end{cases} \\ \text{fir4}[k] &= \delta[k] \end{aligned} \quad (3)$$

where $\delta[k]$ represents the Kronecker delta function. Convolution of the data samples with the filter *fir1* yields the a_i 's coefficients, convolution with the filter *fir2* yields the b_i 's, and so on. Note that the length of the filters (i.e., the number of taps) can be chosen arbitrarily to compromise between computational cost and accuracy.

The second method is based on the cubic B-spline formulation presented by Unser *et al.* [8]. A cubic B-spline is a piece-wise cubic function defined as:

$$\beta^3(t) = \begin{cases} \frac{2}{3} - |t|^2 + \frac{|t|^3}{2} & 0 \leq |t| < 1 \\ \frac{(2-|t|)^3}{6} & 1 \leq |t| < 2 \\ 0 & 2 \leq |t| \end{cases} \quad (4)$$

Using this basis function, we can formulate an analytical representation of the sampled signal $s_1[i]$ as the following convolution sum:

$$s_1(t) = \sum_{i=0}^{N-1} c(i) \beta^3(t - i) \quad (5)$$

where the $c(i)$'s are the B-spline coefficients. Panel A in figure 1 shows a series of discrete points and the corresponding shifted cubic B-spline functions derived from equation (5). While numerous techniques exist to calculate the B-spline coefficients, Unser *et al.* derived an efficient implementation based on 2 infinite impulse response (IIR) filters, each with 2 taps [8]. After a straightforward initialization, the filters perform simple recursion formulas that run through the data from left to right (first, causal filter) and from right to left (second, anti-causal filter). The two filters are given by [8], [11]:

$$\begin{cases} c^+(i) = s(i) + z_1 c^+(i-1) & i = 1 \cdots N-1 \\ c^-(i) = z_1 (c^-(i+1) - c^+(i)) & i = N-2 \cdots 0 \\ c(i) = 6 \cdot c^-(i) & i = 0 \cdots N-1 \end{cases} \quad (6)$$

where $z_1 = -2 + \sqrt{3}$ the first equation is the causal filter, the second equation is the anti-causal filter, and the $c(i)$'s are the B-spline coefficients of equation (5). Once the coefficients are calculated using equation (6), the overlapping segments of the scaled B-splines can be summed over each interval to obtain the piece-wise representation given in equation (1). This is shown schematically in panel A of figure 1, where the spline segments within the shaded box are scaled and summed to generate $f_i(t)$.

The third method described in this paper uses the Catmull-Rom splines [12]. These splines are cubic polynomials formulated such that the tangent at each data point is calculated using the previous and next point on the spline, as shown schematically in panel B of figure 1. The four coefficients for each segment of the spline (i.e., the $f_i(t)$'s) are defined by:

$$\begin{bmatrix} a_i \\ b_i \\ c_i \\ d_i \end{bmatrix} = \frac{1}{2} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} s_1[i-1] \\ s_1[i] \\ s_1[i+1] \\ s_1[i+2] \end{bmatrix} \quad (7)$$

The above equation indicates that each segment is locally defined given four consecutive data points. Furthermore, the same 4 by 4 matrix can be used to derive each of the piece-wise polynomials of equation (1).

2. Solution of the fifth order polynomial

The algorithm described in [6] requires solution of the roots of a 5th order polynomial for every overlap of the delayed signal within the reference. While there are methods to solve those polynomials analytically, using for example the Jacobi theta functions, the complexity of these approaches is such that root finding is most often performed numerically. Thus far we have used a numerical method that is based on finding the eigenvalues of the companion matrix associated with the polynomial analyzed. It has been reported that this algorithm “uses order n^2 storage and order n^3 time” for an n order polynomial [14].

As an alternative, we propose approximating the 5th order polynomial with a 3rd order polynomial, whose coefficients minimize the SSE between the two polynomials over the interval [0 1]. A cubic polynomial represents an excellent compromise between performance and computational cost. Furthermore, it guarantees the existence of at least one real root. We represent the SSE between the 5th order polynomial and its cubic approximation as:

$$\theta = \int_0^1 \left(\frac{\partial \varepsilon}{\partial t} - \frac{\partial \widehat{\varepsilon}}{\partial t} \right)^2 dt \quad (8)$$

where $\frac{d\widehat{\varepsilon}}{dt} = g(t) = \alpha_1 t^3 + \alpha_2 t^2 + \alpha_3 t + \alpha_4$. The 3rd order polynomial coefficients are then found by solving the set of simultaneous equations:

$$\frac{\partial \theta}{\partial \alpha_1} = 0, \quad \frac{\partial \theta}{\partial \alpha_2} = 0, \quad \frac{\partial \theta}{\partial \alpha_3} = 0, \quad \frac{\partial \theta}{\partial \alpha_4} = 0 \quad (9)$$

This yields the following coefficients:

$$\begin{aligned} \alpha_1 &= \sum_{i=1}^M (8a_i c_i + 4b_i^2 + 20a_i b_i + \frac{50}{3}a_i^2) \\ \alpha_2 &= \sum_{i=1}^M (6a_i d_i + 6b_i c_i - 6a_i s_2[i] - \frac{90}{7}a_i b_i - \frac{100}{7}a_i^2) \\ \alpha_3 &= \sum_{i=1}^M (4b_i d_i + 2c_i^2 - 4b_i s_2[i] + \frac{20}{7}a_i b_i + \frac{25}{7}a_i^2) \\ \alpha_4 &= \sum_{i=1}^M (2c_i d_i - 2c_i s_2[i] - \frac{1}{7}a_i b_i - \frac{4}{21}a_i^2) \end{aligned} \quad (10)$$

where a_i , b_i , c_i , and d_i are the spline coefficients of equation (1). The roots of $g(t)$, can be found analytically, significantly reducing computational costs. Alternatively, a companion matrix method can be performed at a reduced computational cost [14].

3. Adaptive search for delay estimate

For every overlap of the kernel within the search, commonly used algorithms such as the normalized correlation and the SSD output a coefficient that indicates the quality of the match for that particular overlap. This is shown schematically in the top panel of figure 2, where two signals are compared to estimate their relative delay. In the case presented in this figure there are 9 possible overlaps between the kernel and the search. The coefficients obtained using normalized correlation, for example, do not offer any information about the location of the “best match” which is indicated by the arrow in this figure. Therefore every overlap must be evaluated to find the maximum of the normalized correlation. Note that in this particular case, the use of post correlation interpolation such as parabolic interpolation, would reveal the true displacement to lie in between overlap 5 and 6 (i.e., between the estimate 0.988 and 0.596).

One of the advantages of the spline-based algorithm presented here is that it is able to predict where the global delay estimate will occur. This is most likely a consequence of the fact that every segment of the spline is dependent upon its neighbor segments. This is shown schematically in the bottom panel of figure 2. The outputs of the spline-based TDE are local time delay estimates that point to the position of the global delay estimate, which is indicated by the arrow in this figure (i.e., 0.8 samples). Using this observation, the speed of the algorithm can be increased substantially by adaptively selecting which overlaps to compute. The first overlap is computed, and the local delay estimate is used to predict where the “best match” will occur. In other words, the local delay estimate will predict how many samples to the right or to the left the best match will occur. A new overlap is then computed at the position indicated by the previous local estimate. This process is repeated until the global delay estimate (or an estimate with a value between [0 1]) is found.

Note that since the Catmull-Rom splines are defined locally, the adaptive search cannot be used in conjunction with this approach.

III. SIMULATED PERFORMANCE

A series of computer simulations were performed to compare the performance of the complete spline-based TDE described in [6] to the modified algorithms presented here. We analyzed the cases when the SSE approximation is used in conjunction with the FIR filters (we will refer to this algorithm as SSE-FIR), the IIR filters (SSE-IIR), and the Catmull-Rom approach (SSE-CR) to determine the spline coefficients. In the case of the SSE-FIR method, we chose a length of 21 taps for the filters in equation (3), corresponding to the arbitrary level of -100dB . Furthermore, the roots of the SSE derived third order polynomial were determined analytically. All calculations were performed in MATLAB (MathWorks Inc., Natick, MA), and utilized synthetic, broadband ultrasound signals. RF echo data were generated convolving a Gaussian enveloped sinusoid with white random noise. The Gaussian enveloped sinusoid had a 5.0MHz center frequency with 50% fractional bandwidth. The kernel length was set to $2\mu\text{sec}$ and the time delay was searched over a $2.2\mu\text{sec}$ window.

Figure 3 depicts the bias and variance of the various algorithms as a function of the sub-sample shift. A signal and its exact shifted replica were used for this case. Under each set of conditions statistics were obtained over an ensemble of 1,000 signal pairs. The top panel of figure 3 shows the absolute value of the measured bias for the algorithms operating on data sampled at 40MHz . The biases of the algorithms follow the expected shape with peaks on the order of 4.5×10^{-4} samples. The SSE-CR shows peaks on the order of 1×10^{-2} samples. The bottom panel of figure 3 depicts the standard deviation of time delay estimates for signals sampled at 40MHz as a function of the sub-sample shift. Both the SSE and the SSE-IIR approaches achieve standard deviations comparable to those of the full spline-based TDE (note that a standard deviation of zero is achievable here because the reference and delayed signals are identical, but for a shift). In the case of the SSE-CR there is a loss of roughly a factor of 10.

A series of simulations were also performed to analyze the effects of the adaptive process as a function of the sub-sample shift. In this case the SSE-FIR approach was used for this set of simulations. Figure 4 shows the results of these simulations for signals sampled at 40MHz . For every sub-sample shift, a series of 1,000 reference and delayed signals were generated. The results presented in this figure show a four-fold decrease in the number of overlaps required to obtain the global delay estimate.

IV. EXPERIMENTAL PERFORMANCE

Experiments were also performed to test the algorithms by acquiring displaced sets of RF echo data using an M3000 mechanical translation stage (Newport Corporation, Mountain View, CA) with a stated precision of $0.1\mu\text{m}$. RF data was acquired at $2.4\mu\text{m}$ intervals in range from within a degassed, open celled sponge placed in a water tank kept at room temperature (22.6°C).

The transducer was mounted to a 1 inch thick Plexiglas plate using nylon cable ties. A two axis tilt and rotation stage (M37 with micrometers, Newport Corporation) was also employed to align the transducer to the movement of the motion stage so as to keep the image plane parallel to the translation plane. Calibration of the alignment system was performed using correlation coefficients obtained before and after an axial translation of 10 samples. The correlation was computed over a 65 lateral by 400 axial sample region after accounting for the sample shift movement so as to compute the correlation on the same region of the phantom at each location. The axial correlation was maximized to 0.99435 after 10 sample shifts.

Echo data was obtained from an Ultrasonix Sonix RP imaging system (Richmond, BC, Canada) with a L14–5/38 linear array transducer. The system speed of sound was adjusted to match that of the water bath (1488m/s). Signals were sampled temporally at 40MHz. Prior to processing, received data was bandpass filtered using a 100th order FIR filter with cutoffs at 2.5 and 8.5MHz. As for the simulations, the kernel length was set to 2 μ sec and delays were searched over a 2.2 μ sec window. For each step of the positioning system, we acquired 50 frames to provide different noise realizations. Each frame was subdivided into 20 non-overlapping, unique speckle patterns. Analyzing each of these speckle patterns across the multiple frames for each displacement, we generated 1,000 trials for each displacement. Figure 5 shows the average over the ensemble of the delay estimates as a function of assumed sub-sample shift; the straight line indicates ideal performance. Note that errorbars for the algorithms are included in this figure. Figure 6 shows the estimated bias (the ideal performance has been subtracted from the estimates) and the standard deviation as a function of assumed sub-sample shift.

V. DISCUSSION AND CONCLUSIONS

We have previously described a spline-based time delay estimator that estimates delays with a high degree of accuracy and precision [6]. In this paper we have presented modified versions of that algorithm that reduce computational cost while maintaining excellent performance. The results presented in figure 3 show that the bias and standard deviation of the SSE-FIR and SSE-IIR are comparable to that of the original spline-based TDE. In the case of the SSE-CR, performance is degraded by a factor of 25 for the bias and roughly a factor of 10 for the standard deviation. Furthermore, if compared with other widely known algorithms, such as normalized correlation and Sum Squared Differences (SSD), the SSE-FIR and the SSE-IIR approaches achieve better performance. Comparison with previous results [6] indicates a bias of about 2.5 orders of magnitude lower than that of the SSD and normalized correlation coupled with parabolic fitting, and 1.5 orders of magnitude lower than those algorithms coupled with Grid Slopes [6]. Note that the bias of the original spline-TDE, as presented in this figure, is larger than the bias presented in [6]. This discrepancy results from an incorrect interpolation scheme in our original paper. This error caused an underestimation of spline-TDE bias in our original paper of a factor between 8 and 10. Figure 3 indicates that at a sub-sample shift of 0.25, the standard deviation is about 11 times smaller than that of the SSD with Grid Slopes interpolation, and about 500 times better than that of normalized correlation with either parabolic or cosine interpolation.

The findings of the experiments are shown in figures 5 and 6. It is noticeable from figure 5 that the algorithms follow very closely the ideal performance represented by the solid line. The SSE-CR algorithm produces estimates that, on average, exhibit larger fluctuation around the ideal line than the other algorithms. This observation is consistent with the results of figure 6. The top panel of figure 6 indicates that the estimated bias of the algorithms is higher than the bias observed in simulations. This is most likely the result of electronic noise and echo signal decorrelation, which were not considered in the simulation methods. However, as observed in figure 3, the performance of the SSE-CR algorithm is inferior to the rest of the algorithms. The bottom panel of figure 6 shows the standard deviation of estimates as a function of assumed sub-sample shift. In this case the algorithms are clustered together and performance decreases following a linear pattern.

The computational complexity of the methods presented in this paper are significantly lower than that required by the original spline-based TDE method. The Catmull-Rom splines can be implemented efficiently with just 3 FIR filters with 4 taps each, corresponding to the first three rows of the matrix in equation (7) (the zero order coefficients are given by the data samples). However, the performance of this method is worse than the other methods. The other FIR based approach presented here uses 3 FIR filters with 21 taps each. Although this is more costly than

the previous approach, the results in figure 3 clearly show the advantages in terms of accuracy and precision. The IIR approach, is implemented using two recursive filters with 2 taps each plus 4 polynomial summations across each sampling interval, as shown schematically in figure 1A. This approach is particularly suitable for a general processor, while the FIR approaches are particularly appropriate when the data is pipelined (i.e., no memory storage for the recursive filters is required).

While finding 3 roots analytically is certainly faster than finding 5 roots numerically, we could improve performance further by solving only for the “correct” third order root. We have found empirically that the value of t that minimizes the SSE is always given by the real part of the same complex root of $g(t)$. This is given by:

$$t_o = \text{Re} \left\{ -\frac{a_2}{3a_1} + \frac{(1-i\sqrt{3})(-a_2^2+3a_1a_3)}{3 \cdot 2^{2/3} a_1 (-2a_2^3+9a_1a_2a_3-27a_1^2a_4 + \sqrt{4(-a_2^2+3a_1a_3)^3 + (-2a_2^3+9a_1a_2a_3-27a_1^2a_4)^2})^{1/3}} \right. \\ \left. - \frac{(1+i\sqrt{3})(-2a_2^3+9a_1a_2a_3-27a_1^2a_4 + \sqrt{4(-a_2^2+3a_1a_3)^3 + (-2a_2^3+9a_1a_2a_3-27a_1^2a_4)^2})^{1/3}}{6 \cdot 2^{1/3} a_1} \right\} \quad (11)$$

Although not presented in this paper, we also analyzed the performance of a third order Taylor series expansion of the fifth order polynomial around 0.5. As expected, this approximation worked very well in the neighborhood of 0.5, but its bias and standard deviation increased rapidly away from that value. Fourth order polynomials were also investigated to approximate the derivative of the error function. However, we found that their performance was equivalent to that of cubic polynomials, but with a much higher computational cost for root finding.

Finally, it is worth noting that the work presented here can be easily extended to two or more dimensions. We have previously presented a spline-based multi-dimensional motion estimator, named MUSE [13]. MUSE uses 1D splines to form a multi-dimensional spline function (typically 2- or 3-D) that represents one data set analytically. Relative motion/shear/compression between the splined and sampled data sets is found by formulating an error function and finding its minimum. Since MUSE is based on the application of 1D splines, the techniques presented here can be used to decrease the computational load of this algorithm, while maintaining excellent performance.

Acknowledgements

We thank Kevin Owen, Drake Guenther, and Ryan Coe for assistance in acquiring the experimental data. We also acknowledge technical support provided by Kris Dickie and Ultrasonix Medical Corporation. The authors would also like to thank the anonymous reviewers who have significantly improved the manuscript. This work was supported by a research grant from the Whitaker Foundation titled “Acoustic Radiation Force Imaging in the Eye” and by Grant R01 EB005433 from NIBIB and the NIH.

References

1. Carter, GC. Coherence and Time Delay Estimation: An Applied Tutorial for Research, Development, Test, and Evaluation Engineers. Piscataway, NJ: IEEE; 1993.
2. IEEE Trans Ultrason, Ferroelect, Freq Contr (Special Issue on Correlation Techniques and Applications to Ultrasonics) 1994;41:577–663.
3. Viola F, Walker WF. A Comparison of the Performance of Time Delay Estimators in Medical Ultrasound. IEEE Trans Ultrason, Ferroelect, Freq Cont 2003;50(4):392–401.
4. Fertner A, Sjölund A. Comparison of Various Time Delay Estimation Methods by Computer Simulation. IEEE Trans on Acous, Speech, Signal Proc 1986;34:1329–1330.
5. Langeland S, D’hooge J, Torp H, Bijnens B, Suetens P. A Simulation Study on the Performance of Different Estimators for Two-Dimensional Velocity Estimation. IEEE Ultrasonic Symposium. 2002
6. Viola F, Walker WF. A Spline Based Algorithm for Continuous Time Delay Estimation Using Sampled Data. IEEE Trans Ultrason, Ferroelect, Freq Cont 2005;52(1):80–93.

7. Spath, H. One dimensional spline interpolation algorithms. A. K. Peters Ltd; 1995.
8. Unser M. Splines: A Perfect Fit for Signal and Image Processing. IEEE Signal Processing Magazine 1999;16(6):22–38.
9. Vrcelj B, Vaidyanathan PP. Efficient Implementation of All-Digital Interpolation. IEEE Trans Image Proc 2001;10(11):1639–1646.
10. Goshtasby A, Cheng F, Barsky BA. B-spline curves and surfaces viewed as digital filters. Computer Vision, Graphics, and Image Processing 1990;52:264–275.
11. Unser M, Aldroubi A, Eden M. B-Spline Signal Processing: Part II – Efficient Design and Applications. IEEE Trans Sig Proc 1993;41(2):834–848.
12. Catmull, E.; Rom, R. A class of local interpolating splines. In: Barnhill, RE.; Reisenfeld, RF., editors. Computer Aided Geometric Design. Academic Press; New York: 1974. p. 317-326.
13. Walker, WF.; Guenther, DA.; Coe, RL.; Viola, F. A Novel Spline-Based Algorithm for Multi-Dimensional Displacement and Strain Estimation; 2006 IEEE Ultrasonics Symposium; Vancouver, Canada.
14. Edelman A, Murakami H. Polynomial Roots From Companion Matrix Eigenvalues. Mathematics of Computation 1995;64(210):763–776.

Figure 1.

Panel A: generation of cubic spline coefficients using B-spline IIR filtering approach. A series of discrete data points are shown as open circles along with the shifted B-spline basis function $\beta(t)$. The scaling coefficients of equation (4) are obtained by filtering the original data samples with 2 IIR filters with 2 taps each. Summation of the scaled basis functions over the shaded area (i.e., over a given sampling interval) yields the spline segment $f_i(t)$.

Panel B: generation of the cubic spline coefficients using the Catmull-Rom spline approach. Given a series of four consecutive sampled data points, a Catmull-Rom spline is formulated by defining the tangent at each data point (i.e., $s_1[i]$ and $s_1[i+1]$) using the previous and next sampled points on the spline (i.e., $s_1[i-1]$ and $s_1[i+2]$). The tangents are shown as gray arrows in this schematic. Note that this approach produces segments of the spline that are locally defined.

Figure 2.

Time delay estimation using the normalized correlation (top) and the spline-based algorithm (bottom). In the case of the normalized correlation, every overlap is computed and a coefficient is output indicating the quality of the match for that particular overlap. Since the individual coefficients hold no information about the location of the “best match” (indicated by the black arrow), the search needs to be computed using the overall sequence of coefficients. In the case of the spline-based estimator, the outputs of the algorithm are local time delay estimates that coherently (albeit with some error as distance increases) point to the position of the global delay estimate, indicated by the black arrow. Each estimate can thus be used to “jump” to the location of the global delay.

Figure 3.

Bias (top panel) and standard deviation (bottom panel) of delay estimates for signals sampled at 40 MHz. Reference and delayed signals were identical except for a sub-sample shift. Each data point is determined from 1,000 trials using simulated 5 MHz, 50% BW speckle signals.

Figure 4.

Comparison between algorithms with and without the adaptive search. Statistics were obtained by generating 1,000 reference and delayed signal sampled at 40 MHz. When the adaptive search is not implemented there are eight possible overlaps between the reference and the delayed signals, as show by the gray bars.

Figure 5.

Mean (solid symbol) and standard deviation (errorbar) of delay estimates from experimental data. RF echo data was obtained from an Ultrasonix RP using a motion stage to provide sub-sample translation of the ultrasound probe. Signals were temporally sampled at 40 MHz. The solid line indicates ideal performance.

Figure 6. Estimated bias (top panel) and standard deviation (bottom panel) of delay estimates from experimental data.