Eurographics Conference on Visualization (EuroVis) 2017
J. Heer, T. Ropinski, and J. van Wijk
(Guest Editors)

Volume 36 (2017), Number 3

Supplemental Material to Global Feature Tracking and Similarity Estimation in Time-Dependent Scalar Fields

H. Saikia and T. Weinkauf

KTH Royal Institute of Technology, Stockholm, Sweden

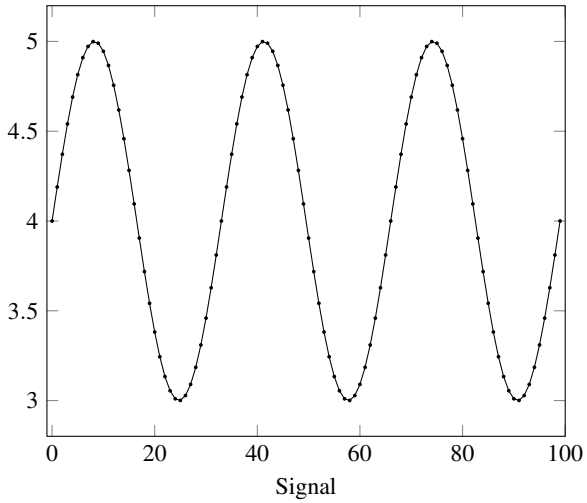

Figure 1: $f(x) = sin(3x)$ *sampled at 100 points.*

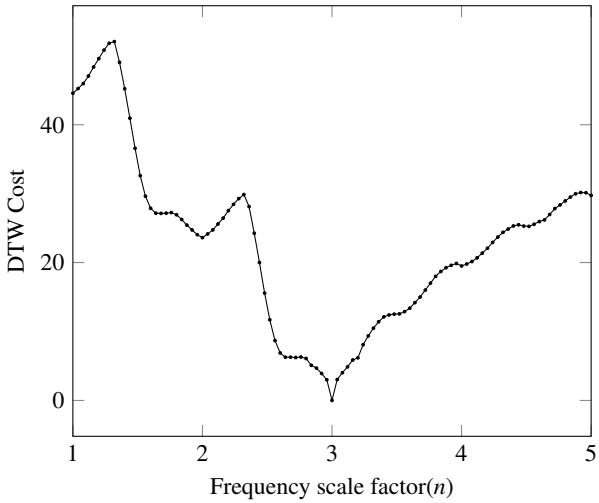

Figure 2: *DTW Costs of comparing a signal* $f(x) = \sin(3x)$ *to a signal* $f(x) = \sin(nx)$.

1. Supplemental Material Overview

We present several examples of a transformed 1D signal and plots of DTW comparison costs for the transformed signal with the original in Section 2. This exemplifies how DTW takes into account both spatial and temporal differences. In Section 3 we elaborate on how the overlap and signature distances are calculated for all subtree pairs in two successive timesteps.

2. Analysis of Dynamic Time Warping behavior on 1D signals

We consider a sine-curve $f(x) = sin(3x)$ sampled at 100 points as our signal. Figure 1 shows this curve. We perform various transformations to this signal and compare these transformations to the original signal using DTW.

Frequency Scaling

Figure 2 shows the plot of DTW costs of comparing the signal $f(x) = \sin(3x)$ to a signal $f(x) = \sin(nx)$ with $n \in [1,5]$. We observe that the cost rises if the frequency is scaled too high or too low, but remains small for frequencies close to the original frequency.

Amplitude Scaling

Figure 3 shows the plot of DTW costs of comparing the signal $f(x) = \sin(3x)$ to a signal $f(x) = a\sin(nx)$ with $a \in [0.5,2]$. We observe that the cost rises almost linearly when we move away from our original signal $a = 1$, as expected.

Phase Shift

Figure 4 shows the plot of DTW costs of comparing the signal $f(x) = \sin(3x)$ to a signal $f(x) = \sin(3x+\phi)$ with $\phi \in [0, 2\pi]$. The DTW cost is highest at $\phi = \pi$ when the two signals are completely out of phase. At $\phi = 2\pi$ the signals become identical again and the cost goes down to zero. The jumps in the curve are observed because of the chosen sampling distance.

Uniform Noise

We tamper our signal by adding some uniform noise to every sample. We compare 100 such noisy signals to the original one at increasing noise levels and plot the average of all 100 costs. Figure 5 shows these observations. Our noise level $\ell \in [0, 0.1]$ is based on

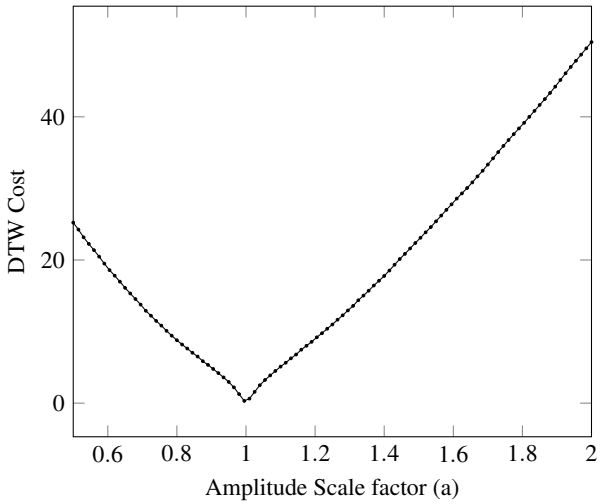

Figure 3: *Comparison of* $f(x) = \sin(3x)$ *to* $f(x) = a\sin(3x)$ *where* $a \in [0.5, 2]$

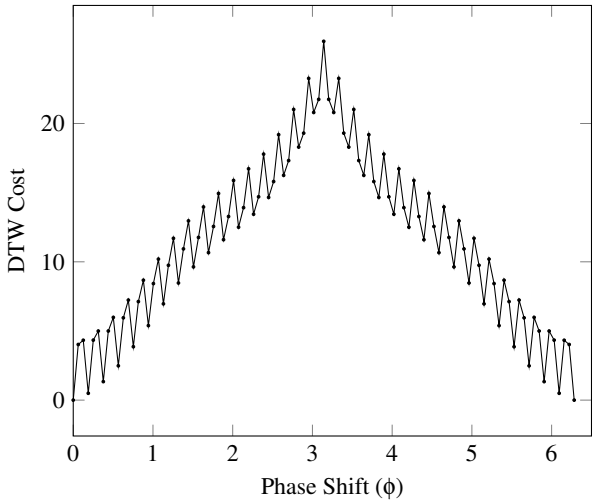

Figure 4: *Comparison of* $f(x) = \sin(3x)$ *to* $f(x) = \sin(3x+\phi)$ *with* $\phi \in [0, 2\pi]$.

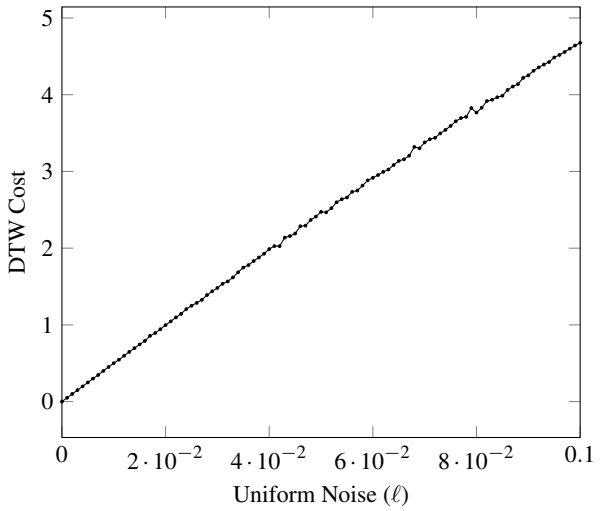

Figure 5: *Comparison of* $f(x) = \sin(3x)$ *to* $f(x) = (1 + rand(-\ell, \ell))\sin(3x)$ *where* $\ell \in [0, 0.1]$

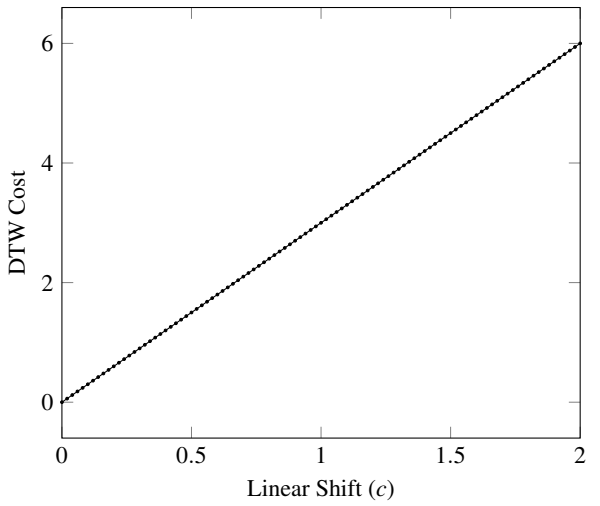

Figure 6: *Cost of comparing a linear signal* $f(x) = 3 + 2x$, $x \in [0, 1]$ *with a linearly shifted signal* $f(x) = (3+c) + (2-2c)x$ *where* $c \in [0, 2]$

the % change in amplitude of the signal. As expected, the DTW cost increases with increasing noise levels.

Linear shift

For this experiment we consider a linear signal $f(x) = 3 + 2x$, $x \in [0, 1]$ and transform it to the signal $f(x) = 5 - 2x$, $x \in [0, 1]$, by shifting the end-point $x = 0, y = 3$ upwards and the end-point $x = 1, y = 5$ downwards. After shifting by 2 units, we arrive at the second signal. We do this in 100 steps. We compare the original signal to the shifted signal after every step. The results of the comparison are shown in Figure 6. This plot is as expected, i.e., with every linear shift, the L1 distance between each corresponding sample point increases linearly, hence an overall linear increase in DTW cost.

3. Distance Computation between two subtrees

In the following subsections we elaborate on the computation of overlap as well as the signature difference, between all subtrees of two successive timesteps.

3.1. Computing Overlap

Consider two merge trees M_1 and M_2, obtained from scalar fields f_1 and f_2 defined in the same domain $\mathscr{D}$. We are interested in finding the overlap of all subtrees in M_1 with all subtrees in M_2. Figure 7 shows a merge tree with 7 edges. Hence, there are 7 subtrees. Each subtree can be uniquely represented by an edge. The subtree, $AE - BE - EG$ for example, can be represented by the edge EG, or

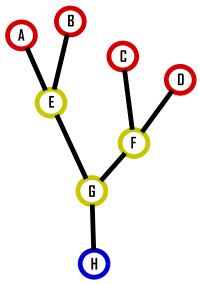

Figure 7: *A merge tree*

its lowermost edge. We denote this edge by $\mathcal{E}_{EG}$ or simply by $\mathcal{E}_E$ since every edge is also uniquely represented by its upper node.

Since a subtree consists of smaller subtrees, it can be represented as a collection of an edge and several smaller subtrees. Using the notation from before we can write $\mathcal{S}_E = \mathcal{E}_E \cup \mathcal{S}_A \cup \mathcal{S}_B$. A subtree with only one edge is simply given by its edge i.e. $\mathcal{S}_A = \mathcal{E}_A$. Hence in general we can write

$$\mathcal{S}_i = \mathcal{E}_i \cup (\bigcup_{x \in C_i} \mathcal{S}_x) \tag{1}$$

where C_i is the set of all child subtrees connected to $\mathcal{E}_i$.

From the merge tree computation, we know for every voxel $v \in \mathscr{D}$ which edge of the merge tree it belongs to. Let this function be called g. Hence $\mathcal{E}_1 \cup \mathcal{E}_2 \cup \ldots \cup \mathcal{E}_n = \mathscr{D}$ for any merge tree with n edges. The overlap between two edges $\mathcal{E}_{1,i} \in M_1$ and $\mathcal{E}_{2,j} \in M_2$ can hence be computed quite easily by looking at all voxels $v \in \mathscr{D}$ such that $g_1(v) = \mathcal{E}_{1,i}$ and $g_2(v) = \mathcal{E}_{2,j}$. This can be done in $O(N)$ time where N is the size of the dataset. Hence the overlap between two edges is given by

$$\mathcal{E}_{1,i} \cap \mathcal{E}_{2,j} = \{v : g_1(v) = \mathcal{E}_{1,i} \wedge g_2(v) = \mathcal{E}_{2,j}, v \in \mathscr{D}\} \tag{2}$$

The overlap between two subtrees can be written as

$$\begin{aligned} \mathcal{S}_{1,i} \cap \mathcal{S}_{2,j} = {} & (\mathcal{E}_{1,i} \cap \mathcal{E}_{2,j}) \\ & \cup (\mathcal{E}_{1,i} \cap \mathcal{S}_{2,j}) \\ & \cup (\mathcal{S}_{1,i} \cap \mathcal{E}_{2,j}) \\ & \cup (\bigcup_{x \in C_{1,i}, y \in C_{2,j}} \mathcal{S}_x \cap \mathcal{S}_y) \end{aligned} \tag{3}$$

which is a combination of 3 recursive functions. These are easily solved by decomposing every subtree into its constituent edge and child subtrees and using the precomputed values for every pair of edges as computed using Equation 2. Computed scores are memoized to avoid recomputing in a different recursive instance. The overlap between two subtrees is now given recursively as the sum of the overlap between their constituent edges, their child subtrees, and combinations of the two. These recursions are solved in approximately $O(m \cdot n)$ time using memoization for two timesteps with $m \times n$ subtrees.

3.2. Computing Signature Distance

We use histograms of voxel intensities as our spatial comparison signatures. Comparing two histograms of B buckets each using χ^2-distance can be done in $O(B)$ time. Hence, for two timesteps with $m \times n$ subtrees, signature distances can be computed in $O(m \cdot n \cdot B)$ time.