Tanmay Mishra
(630)-854-8089
tmishra2003@gmail.com

Ableton Programming Assignment #1
01/28/2019

Introduction and Analysis

When working with samples in audio software, the user should have the flexibility to manipulate the sample as he/she pleases. A commonly used operation, known as warping (time stretching), allows the user to stretch/shrink any portion of the sample. By defining points in the sample (known as warp markers) to designate the start of a new tempo section, the user can create as many warp markers as they need to, creating as many stretch/shrinks as needed.

We abstract a time warp marker as a mapping between time locations in the original sample to its corresponding location in the sequencer in beat units. The mapping is what allows us to refer to fixed points between the original sample and the warped versions.

We can denote a warp marker as a pair, $w = (B, T)$[1], where B is the beat value in the sequencer and T is the corresponding time value in the original audio sample.

[1] Because the tempo is B/T, having B in the pair first is more consistent but not intuitively appropriate.

As warping changes the length of a clip in the sequencer, we need a way to map points on the sample to the actual, shifted time values. This function takes in a value of a particular time on the sample or takes in a particular beat value on the sequencer and returns the corresponding value in beat units or seconds.

As we warp different sections, we are effectively changing the tempo. Tempo is defined as beats divided by time (B/T). By measuring the change in beats over its corresponding change in time, we can measure the tempo of that region. We will leverage this methodology.

Context/Constraints

In the scope of this problem,

1. Tempo is defined as beats/time.
2. W denotes a set of warp markers and contains at least 1 warp marker. That is, $n \geq 1$ where n is the number of warp markers.
3. We need a user defined tempo after the last warp marker. This is referred as R_{end}.
4. All input values are assumed to be valid.

Problem

1. Given B_u , the user inputted beat value, find T_u , the corresponding time value.
2. Given T_u , the user inputted time value, find B_u , the corresponding beat value.

Approach and Solution

We represent our set of warp markers, W , as an std::vector object in our C++ interpretation. Each time a warp marker is added/removed, we sort our vector in ascending order.

Mathematically,
W = $\{(B_0, T_0), (B_1, T_1), (B_1, T_1), ..., (B_n, T_n)\}$

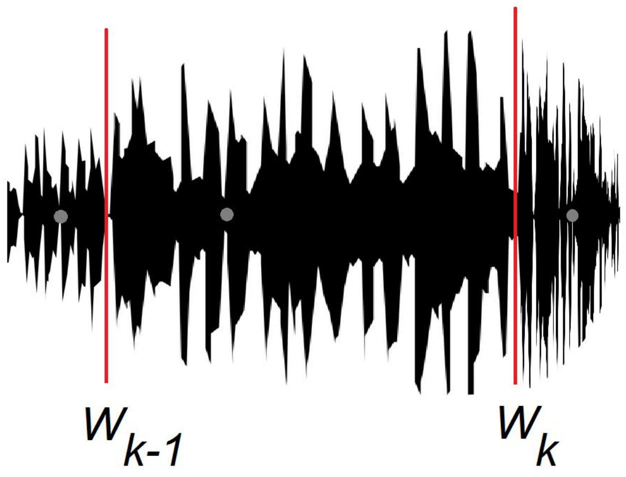
And

W_k = (B_k , T_k)

$$1 \leq k \leq n$$

$$n = number\ of\ warp\ markers$$

As the warp markers adjust tempo, we need to be able to relate warp markers to tempo values. The tempo, $R_{[k-1:k]}$, between two adjacent warp markers, W_{k-1} and W_k, with is calculated as the change in beats over the change in time.



$$R_{[k-1:k]} = \frac{B_k - B_{k-1}}{T_k - T_{k-1}}$$

Given our input value of B_u or T_u and n warp markers, we have 3 different scenarios:

The given value of B or T resides:

1. between 2 warp markers, W_{k-1} and W_k.
2. between the beginning of the clip and the 1st warp marker, W_k.
3. between the nth warp marker, W_n, and the end of the clip.

Scenario 1

Given that our value of (B_u, T_u) lies in between W_{k-1} and W_k,

We know that

$$B_{k-1} < B_u < B_k \text{ and } T_{k-1} < T_u < T_k .$$

Therefore, we can derive

$$\begin{aligned} B_u &= R_{[k-1:k]} \cdot (T_u - T_{k-1}) + B_{k-1} \\ &= \tfrac{B_k - B_{k-1}}{T_k - T_{k-1}} \cdot (T_u - T_{k-1}) + B_{k-1} \quad \textbf{(1)} \end{aligned}$$

And

$$\begin{aligned} T_u &= (1/R_{[k-1:k]}) \cdot (B_u - B_{k-1}) + T_{k-1} \\ &= \tfrac{T_k - T_{k-1}}{B_k - B_{k-1}} \cdot (B_u - B_{k-1}) + T_{k-1} \quad \textbf{(2)} \end{aligned}$$

Scenario 2

Given that our value of (B_u, T_u) lies in between the beginning of the audio sample and W_k **,** we can assume that the beginning of the audio sample is a warp marker at point $(0, 0)$. If we assume this to be W_{k-1}, we can say that our corresponding point lies between $W_{k-1} = (0, 0)$ and W_k **,** the first warp marker**.**

Therefore, with the equation from scenario 1, we can derive:

$$B_u = R_{[k-1:k]} \cdot (T_u - T_{k-1}) + B_{k-1} \text{ but } W_{k-1} = (0, 0) \text{ so}$$

$$B_u = R_{[k-1:k]} \cdot T_u \text{ and } R_{[k-1:k]} = R_k = \frac{B_k - B_{k-1}}{T_k - T_{k-1}} = \frac{B_k}{T_k}$$

So

$$B_u = \frac{B_k}{T_k} \cdot T_u \quad \textbf{(3)}$$

Similarly,

$$T_u = (1/R_{[k-1:k]}) \cdot (B_u - B_{k-1}) + T_{k-1}$$

$$T_u = (1/R_{[k-1:k]}) \cdot B_u \text{ and } R_{[k-1:k]} = R_k = \frac{B_k - B_{k-1}}{T_k - T_{k-1}} = \frac{B_k}{T_k}$$

$$T_u = \frac{T_k}{B_k} \cdot B_u \quad \textbf{(4)}$$

Scenario 3

Given that our value of (B_u, T_u) lies after the last warp marker, we can assume the last warp marker to be W_{k-1} with $k - 1 = n$. We also know that the tempo of the audio clip after the last warp marker is R_{end}.

But, what do we do about the value of W_k? We don't need to worry about it because we only need it to calculate the value of $R_{[k-1:k]}$ which we are given because the tempo after the last warp marker *is* what $R_{[k-1:k]}$ would solve for.

So, we can move forward and say:

$$B_u = R_{[k-1:k]} \cdot (T_u - T_{k-1}) + B_{k-1}$$

but $R_{[k-1:k]} = R_{end}$ and $k - 1 = n$

so

$$B_u = R_{end} \cdot (T_u - T_n) + B_n \quad \textbf{(5)}$$

And

$$T_u = (1/R_{end}) \cdot (B_u - B_n) + T_n \quad \textbf{(6)}$$

Now that we've solved for all 3 scenarios, we're going to move on to assembling this into software.

Algorithm Approach

Approaching this from a top down perspective, we need to carry through a couple steps assuming we have at least 1 warp marker and the value of the tempo after the last warp marker:

1. Because our vector of warp markers is sorted in ascending order,
   a. If $B_1 < B_u \leq B_n$ or $T_1 < T_u \leq T_n$, search for value of k for which $B_{k-1} < B_u \leq B_k$ or $T_{k-1} < T_u \leq T_k$.

      Store values of (B_{k-1}, T_{k-1}) as W_{k-1} and (B_k, T_k) as W_k.

      Skip to step d.

   b. If $0 \leq B_u \leq B_1$ or $0 \leq T_u \leq T_1$,

      store values of $(0, 0)$ as W_{k-1} and (B_1, T_1) as W_k.

Skip to step d.

c. If $B_n < B_u$ or $T_n < T_u$, store values of (B_n, T_n) as W_{k-1} and skip directly to storing $R_{[k-1:k]} = R_{end}$ from the given user input.

Skip to step d.

d. Run equation **(1)** or **(2)** with the given input values. Remember that some inputs don’t work and we work around them (Scenario 3). Return the output value.