Subject: Audio Compression Idea
Posted by Kim on Thu, 10 May 2007 07:20:42 GMT
View Forum Message <> Reply to Message

I'm not sure if I'm missing something here, but if I am I don't see what it could be. I'm just wondering what the point is of a WAV file given that they are so big and clunky. I was considering different options for storing audio. This particular concept is a bit of a mix of the concept of floating point numbers, mixed with the SA audio format, in combination with some other ideas. I came to the conclusion that you could store audio in a far better compressed and lossless format, with infinite amplitude, using far less bits than standard WAV files, with an end result identical in accuracy to a standard WAV file except without the amplitude limit.

Programmers amongst you, ponder this for a moment:

Imagine that your audio packets are sequenced in binary like so:

Bit 1: +/- bit - signifies whether the wave is going up or down.
Subsequent data: XY pair where X is a data bit, and Y is the end bit.

....continue with said pattern until an end bit is found.

For example, silence, or any flat signal, would be represented with the sequence 001 repeated. The first zero is the +/- sign (in this case irrelevant as the data is a zero). The second bit is the signal movement data (0 indicates the amplitude hasn't moved) and the last bit is the end bit, indicating we can move on to the next piece of data. This would create a flat line in 3 bits, rather than 16 bits.

If the signal is rising or falling by 45 degrees or less, then we can still represent it in 3 bits. The 3 bits allows the signal to rise in steps of a single bit, hence allowing you to map an exact digital path of up to 45 degrees in either direction with three bits instead of 16.

If you go through the math (assuming a 100& value is a straight line directly up and 0% is directly across):

Rising by 75% would use 5 bits
Rising by 87.5% would use 7 bits.
15 bits would allow a rise/fall of 99.21875%

I can't imagine it would be very often that you would require a rise or fall of greater than 99.21875%, hence you would always be saving space. Worst case, a 33 bit message would give you exactly the same ability to rise and fall as the current 16 bit wav message gets if you go from 0 to 65535 in one step, but I can't imagine this ever happens in reality. The point is you could replicate it if required.

You could gain further efficiency from the first two bits by slightly changing the system. So:

001 = No change
011 = Subtract 1 from value
101 = Add 1 to value
111 = Add 2 to value

By doing this a three bit packet can go anywhere from down 45 degrees, to up 67.5 degrees. You could gain even more data efficiency by allowing a "change direction" of the bias in the header packet, so if we had a 1KB number series, the initial packet would have a bit signifying whether the bias in this packet was towards up, or down. Using this as a system we could probably reduce the vast amount of audio down to just 3 bit packets, rather than 16 bit packets.

You would have:
(*) Infinite amplitude to your signal
(*) Lossless compression
(*) Probably 75% or more reduction in data size compared to wav.

You could use near identical technology with video. Each bit of video data could be represented in 3 bits assuming that they were not changing in value or were changing slowly, and once again you would have lossless compression.

I have another theory to gain a more analog sound, which I'm still considering, but I thought I'd throw that one out there. Some comments would be good, and if somebody with programming nouse would like to throw an encoder/decoder together, well go for it, so long as the format stays in the public domain.

What do you think? Sound feasible?

Cheers,
Kim.

Subject: Re: Audio Compression Idea
Posted by TCB on Thu, 10 May 2007 21:29:05 GMT
View Forum Message <> Reply to Message

Interesting stuff. Before you go too much further you might want to check out the source to FLAC, a GPL'd lossless codec.

http://flac.sourceforge.net/

I use it myself.

TCB

"Kim" <hiddensounds@hotmail.com> wrote:
>
>
>I'm not sure if I'm missing something here, but if I am I don't see what
>it could be. I'm just wondering what the point is of a WAV file given that
>they are so big and clunky. I was considering different options for storing
>audio. This particular concept is a bit of a mix of the concept of floating
>point numbers, mixed with the SA audio format, in combination with some
other
>ideas. I came to the conclusion that you could store audio in a far better
>compressed and lossless format, with infinite amplitude, using far less
bits
>than standard WAV files, with an end result identical in accuracy to a standard
>WAV file except without the amplitude limit.
>
>Programmers amongst you, ponder this for a moment:
>
>Imagine that your audio packets are sequenced in binary like so:
>
>Bit 1: +/- bit - signifies whether the wave is going up or down.
>Subsequent data: XY pair where X is a data bit, and Y is the end bit.
>
>...continue with said pattern until an end bit is found.
>
>For example, silence, or any flat signal, would be represented with the
sequence
>001 repeated. The first zero is the +/- sign (in this case irrelevant as
>the data is a zero). The second bit is the signal movement data (0 indicates
>the amplitude hasn't moved) and the last bit is the end bit, indicating
we
>can move on to the next piece of data. This would create a flat line in
3
>bits, rather than 16 bits.
>
>If the signal is rising or falling by 45 degrees or less, then we can still
>represent it in 3 bits. The 3 bits allows the signal to rise in steps of
>a single bit, hence allowing you to map an exact digital path of up to 45
>degrees in either direction with three bits instead of 16.
>
>If you go through the math (assuming a 100& value is a straight line directly
>up and 0% is directly across):
>
>Rising by 75% would use 5 bits
>Rising by 87.5% would use 7 bits.
>15 bits would allow a rise/fall of 99.21875%
>
>I can't imagine it would be very often that you would require a rise or

fall
>of greater than 99.21875%, hence you would always be saving space. Worst
>case, a 33 bit message would give you exactly the same ability to rise and
>fall as the current 16 bit wav message gets if you go from 0 to 65535 in
>one step, but I can't imagine this ever happens in reality. The point is
>you could replicate it if required.
>
>You could gain further efficiency from the first two bits by slightly changing
>the system. So:
>
>001 = No change
>011 = Subtract 1 from value
>101 = Add 1 to value
>111 = Add 2 to value
>
>By doing this a three bit packet can go anywhere from down 45 degrees, to
>up 67.5 degrees. You could gain even more data efficiency by allowing a
"change
>direction" of the bias in the header packet, so if we had a 1KB number series,
>the initial packet would have a bit signifying whether the bias in this
packet
>was towards up, or down. Using this as a system we could probably reduce
>the vast amount of audio down to just 3 bit packets, rather than 16 bit
packets.
>
>You would have:
>(*) Infinite amplitude to your signal
>(*) Lossless compression
>(*) Probably 75% or more reduction in data size compared to wav.
>
>You could use near identical technology with video. Each bit of video data
>could be represented in 3 bits assuming that they were not changing in value
>or were changing slowly, and once again you would have lossless compression.
>
>I have another theory to gain a more analog sound, which I'm still considering,
>but I thought I'd throw that one out there. Some comments would be good,
>and if somebody with programming nouse would like to throw an encoder/decoder
>together, well go for it, so long as the format stays in the public domain.
>
>What do you think? Sound feasible?
>
>Cheers,
>Kim.

Subject: Re: Audio Compression Idea
Posted by Paul Artola on Fri, 11 May 2007 03:56:17 GMT

It's been too long a day for me to really wrap my brain around your proposal, but it does sound like you are proposing something like adaptive differential pulse code modulation (ADPCM). What you describe is a differential encoding stream - reporting the change from state to state versus actually reporting the state. Provided your underlying data stream is error free, differential encoding makes sense if you are modelling a system with gradually changing codewords.

Alternatively, I wonder if some type of Huffman code (using variable length codewords - shortest for the most frequent "characters" to longest for the least) might work too.

Sorry to get technical here, but I happen to work with some of the world's experts on encoding data. My first boss was a student of the B in BCH codes.

- Paul Artola
  Ellicott City, Maryland

On 10 May 2007 17:20:42 +1000, "Kim" <hiddensounds@hotmail.com> wrote:

>
>
>I'm not sure if I'm missing something here, but if I am I don't see what
>it could be. I'm just wondering what the point is of a WAV file given that
>they are so big and clunky. I was considering different options for storing
>audio. This particular concept is a bit of a mix of the concept of floating
>point numbers, mixed with the SA audio format, in combination with some other
>ideas. I came to the conclusion that you could store audio in a far better
>compressed and lossless format, with infinite amplitude, using far less bits
>than standard WAV files, with an end result identical in accuracy to a standard
>WAV file except without the amplitude limit.
>
>Programmers amongst you, ponder this for a moment:
>
>Imagine that your audio packets are sequenced in binary like so:
>
>Bit 1: +/- bit - signifies whether the wave is going up or down.
>Subsequent data: XY pair where X is a data bit, and Y is the end bit.
>
>...continue with said pattern until an end bit is found.
>
>For example, silence, or any flat signal, would be represented with the sequence
>001 repeated. The first zero is the +/- sign (in this case irrelevant as
>the data is a zero). The second bit is the signal movement data (0 indicates
>the amplitude hasn't moved) and the last bit is the end bit, indicating we
>can move on to the next piece of data. This would create a flat line in 3

>bits, rather than 16 bits.
>
>If the signal is rising or falling by 45 degrees or less, then we can still
>represent it in 3 bits. The 3 bits allows the signal to rise in steps of
>a single bit, hence allowing you to map an exact digital path of up to 45
>degrees in either direction with three bits instead of 16.
>
>If you go through the math (assuming a 100& value is a straight line directly
>up and 0% is directly across):
>
>Rising by 75% would use 5 bits
>Rising by 87.5% would use 7 bits.
>15 bits would allow a rise/fall of 99.21875%
>
>I can't imagine it would be very often that you would require a rise or fall
>of greater than 99.21875%, hence you would always be saving space. Worst
>case, a 33 bit message would give you exactly the same ability to rise and
>fall as the current 16 bit wav message gets if you go from 0 to 65535 in
>one step, but I can't imagine this ever happens in reality. The point is
>you could replicate it if required.
>
>You could gain further efficiency from the first two bits by slightly changing
>the system. So:
>
>001 = No change
>011 = Subtract 1 from value
>101 = Add 1 to value
>111 = Add 2 to value
>
>By doing this a three bit packet can go anywhere from down 45 degrees, to
>up 67.5 degrees. You could gain even more data efficiency by allowing a "change
>direction" of the bias in the header packet, so if we had a 1KB number series,
>the initial packet would have a bit signifying whether the bias in this packet
>was towards up, or down. Using this as a system we could probably reduce
>the vast amount of audio down to just 3 bit packets, rather than 16 bit packets.
>
>You would have:
>(*) Infinite amplitude to your signal
>(*) Lossless compression
>(*) Probably 75% or more reduction in data size compared to wav.
>
>You could use near identical technology with video. Each bit of video data
>could be represented in 3 bits assuming that they were not changing in value
>or were changing slowly, and once again you would have lossless compression.
>
>I have another theory to gain a more analog sound, which I'm still considering,
>but I thought I'd throw that one out there. Some comments would be good,
>and if somebody with programming nouse would like to throw an encoder/decoder

>together, well go for it, so long as the format stays in the public domain.
>
>What do you think? Sound feasible?
>
>Cheers,
>Kim.

Subject: Re: Audio Compression Idea
Posted by Kim on Fri, 11 May 2007 05:34:10 GMT
View Forum Message <> Reply to Message

Paul,

No need to apologise for getting technical. It sounds interesting. I'm probably behind on what the technology is which is out there. I guess I was just surprised when I estimated such a potential compression in a lossless format.

I have another "lossy" compression idea also. No doubt it's an idea that already exists also. ;o) It never hurts to go through such things though.

Cheers,
Kim.

Paul Artola <artola@comcast.net> wrote:
>It's been too long a day for me to really wrap my brain around your
>proposal, but it does sound like you are proposing something like
>adaptive differential pulse code modulation (ADPCM). What you describe
>is a differential encoding stream - reporting the change from state to
>state versus actually reporting the state. Provided your underlying
>data stream is error free, differential encoding makes sense if you
>are modelling a system with gradually changing codewords.
>
>Alternatively, I wonder if some type of Huffman code (using variable
>length codewords - shortest for the most frequent "characters" to
>longest for the least) might work too.
>
>Sorry to get technical here, but I happen to work with some of the
>world's experts on encoding data. My first boss was a student of the B
>in BCH codes.
>
>- Paul Artola
> Ellicott City, Maryland
>
>On 10 May 2007 17:20:42 +1000, "Kim" <hiddensounds@hotmail.com> wrote:
>
>>
>>

>>I'm not sure if I'm missing something here, but if I am I don't see what
>>it could be. I'm just wondering what the point is of a WAV file given that
>>they are so big and clunky. I was considering different options for storing
>>audio. This particular concept is a bit of a mix of the concept of floating
>>point numbers, mixed with the SA audio format, in combination with some
other
>>ideas. I came to the conclusion that you could store audio in a far better
>>compressed and lossless format, with infinite amplitude, using far less
bits
>>than standard WAV files, with an end result identical in accuracy to a
standard
>>WAV file except without the amplitude limit.
>>
>>Programmers amongst you, ponder this for a moment:
>>
>>Imagine that your audio packets are sequenced in binary like so:
>>
>>Bit 1: +/- bit - signifies whether the wave is going up or down.
>>Subsequent data: XY pair where X is a data bit, and Y is the end bit.
>>
>>...continue with said pattern until an end bit is found.
>>
>>For example, silence, or any flat signal, would be represented with the
sequence
>>001 repeated. The first zero is the +/- sign (in this case irrelevant as
>>the data is a zero). The second bit is the signal movement data (0 indicates
>>the amplitude hasn't moved) and the last bit is the end bit, indicating
we
>>can move on to the next piece of data. This would create a flat line in
3
>>bits, rather than 16 bits.
>>
>>If the signal is rising or falling by 45 degrees or less, then we can still
>>represent it in 3 bits. The 3 bits allows the signal to rise in steps of
>>a single bit, hence allowing you to map an exact digital path of up to
45
>>degrees in either direction with three bits instead of 16.
>>
>>If you go through the math (assuming a 100& value is a straight line directly
>>up and 0% is directly across):
>>
>>Rising by 75% would use 5 bits
>>Rising by 87.5% would use 7 bits.
>>15 bits would allow a rise/fall of 99.21875%
>>
>>I can't imagine it would be very often that you would require a rise or
fall
>>of greater than 99.21875%, hence you would always be saving space. Worst

>>case, a 33 bit message would give you exactly the same ability to rise
and
>>fall as the current 16 bit wav message gets if you go from 0 to 65535 in
>>one step, but I can't imagine this ever happens in reality. The point is
>>you could replicate it if required.
>>
>>You could gain further efficiency from the first two bits by slightly changing
>>the system. So:
>>
>>001 = No change
>>011 = Subtract 1 from value
>>101 = Add 1 to value
>>111 = Add 2 to value
>>
>>By doing this a three bit packet can go anywhere from down 45 degrees,
to
>>up 67.5 degrees. You could gain even more data efficiency by allowing a
"change
>>direction" of the bias in the header packet, so if we had a 1KB number
series,
>>the initial packet would have a bit signifying whether the bias in this
packet
>>was towards up, or down. Using this as a system we could probably reduce
>>the vast amount of audio down to just 3 bit packets, rather than 16 bit
packets.
>>
>>You would have:
>>(*) Infinite amplitude to your signal
>>(*) Lossless compression
>>(*) Probably 75% or more reduction in data size compared to wav.
>>
>>You could use near identical technology with video. Each bit of video data
>>could be represented in 3 bits assuming that they were not changing in
value
>>or were changing slowly, and once again you would have lossless compression.
>>
>>I have another theory to gain a more analog sound, which I'm still considering,
>>but I thought I'd throw that one out there. Some comments would be good,
>>and if somebody with programming nouse would like to throw an encoder/decoder
>>together, well go for it, so long as the format stays in the public domain.
>>
>>What do you think? Sound feasible?
>>
>>Cheers,
>>Kim.
>