Frame Context Selection

“TEMPMV_SIGNALING but for frame_context”

Thomas Daede
tdaede@mozilla.com
April 12, 2017

Motivation

Proposals to improve reference frame identification have already been adopted by AV1, for example the REFERENCE_BUFFER experiment. This explicitly identifies which previously coded frames are required to code the current frame, which is required for temporal scalability and loss resilience. However, there are still remaining dependencies not described by the current experiment. This document describes a proposal to remove all remaining untracked dependencies.

Implementation

Currently in AV1, the “frame context” feature has four slots to save probabilities. Each frame can choose which of the four slots to save probabilities. Subsequent frames can load probabilities from any of the four slots.

We remove the four frame contexts, and instead assign one frame context per reference buffer. The syntax elements for saving frame contexts are removed – frame contexts are always saved alongside the reference buffer. Syntax for selecting frame contexts is also removed – the context associated with the first specified reference frame (LAST_FRAME) is used.

Drawbacks

By having one frame context per reference buffer, we increase the number of stored contexts from 4 to 8. However, because only one context is actively loaded per frame, and contexts are much smaller than the associated reference frame, the impact is expected to be minimal.

Specification changes:

1.1.1 Refresh probs semantics

load_probs(~~ctx~~) is a function call that indicates that the probability tables are loaded from ~~frame context number ctx in the range 0 to 3.~~ from the frame indicated by the ref_frame_idx[LAST_FRAME], or reference frame 0 if there are no reference frames. When this function is invoked the following takes place:

- A copy of each probability table (except tx_probs and skipProb) is loaded from ~~an area of memory~~ the reference frame indexed by ref_frame_idx[LAST_FRAME], or reference frame 0 if there are no reference frames. ~~(The memory contents of these frame contexts have been initialized by previous calls to save_probs).~~

load_probs2(~~ctx~~) is a function call that indicates that the probability tables tx_probs and skipProb are loaded from ~~frame context number ctx in the range 0 to 3.~~ from the frame indicated by the ref_frame_idx[LAST_FRAME], or reference frame 0 if there are no reference frames. When this function is invoked the following takes place:

- A copy of the probability tables called tx_probs and skipProb are loaded from ~~an area of memory~~ the reference frame indexed by ref_frame_idx[LAST_FRAME], or reference frame 0 if there are no reference frames.

1.2 Uncompressed header semantics

~~**frame_context_idx** indicates the frame context to use.~~

~~**save_probs(ctx)** is a function call that indicates that indicates that all the probability tables are saved into frame context number ctx in the range 0 to 3. When this function is invoked the following takes place:~~

~~A copy of each probability table is saved in an area of memory indexed by ctx. The memory contents of these frame contexts are persistent in order to allow a subsequent inter frame to reload the probability tables.~~

save_probs(flags) is a function call that indicates that indicates that all the probability tables are saved into frame contexts associated with the reference frames specified by flags. This much always match referesh_frame_flags except when resetting all frames (0xFF) When this function is invoked the following takes place:

A copy of each probability table is saved alongside the corresponding reference frame. The memory contents of these frame contexts are persistent in order to allow a subsequent inter frame to reload the probability tables.

~~**refresh_frame_context** equal to 1 indicates that the probabilities computed for this frame (after adapting to the observed frequencies if adaption is enabled) are stored for reference by future frames. refresh_frame_context equal to 0 indicates that the probabilities are discarded at the end of the frame.~~

1.1 Frame order constraints

- When load_probs(~~ctx~~) is invoked, this must load an initialized set of probabilities~~, i.e. there must have been an earlier invocation of save_probs(ctx).~~ i.e. it must reference a valid previous reference frame. If there are no reference frames, it will read from reference frame 0, which must have been initialized to the default probabilities previously with setup_past_independence().

1.2.1 Refresh probs function

refresh_probs( ) {	**Type**
if (error_resilient_mode == 0) {	
load_probs(~~ frame_context_idx~~)	
adapt_coef_probs( )	
if (FrameIsIntra == 0) {	
load_probs2(~~ frame_context_idx~~)	
adapt_noncoef_probs( )	
}	
}	
~~if (refresh_frame_context)~~	
save_probs(~~frame_context_idx~~ refresh_frame_flags)	
~~}~~	

1.3 Uncompressed header syntax

~~if (error_resilient_mode == 0) {~~
~~**refresh_frame_context**~~
~~} else {~~
~~refresh_frame_context = 0~~
~~}~~
~~**frame_context_idx**~~
if (FrameIsIntra || error_resilient_mode) {
setup_past_independence ( )
if (frame_type == KEY_FRAME || error_resilient_mode == 1 || ResetFrameContext == 2) {
~~for (i = 0; i < 4; i ++) {~~
save_probs(~~i~~ 0xFF)
~~}~~
} else if (ResetFrameContext == 1) {
save_probs(~~frame_context_idx~~ refresh_frame_flags)
}
~~frame_context_idx = 0~~
}