A Technical Overview of VP9:
The latest royalty-free video codec from Google

Google™

Debargha Mukherjee

Team: Jim Bankoski, Ronald S. Bultje, Adrian Grange,
Jingning Han, John Koleszar, Debargha Mukherjee,
Yunqing Wang, Paul Wilkins, Yaowu Xu

Outline

- Introduction
- VP9 Bit-stream overview
  - Coding Tools
  - Bitstream Features
- Coding results
- Conclusion

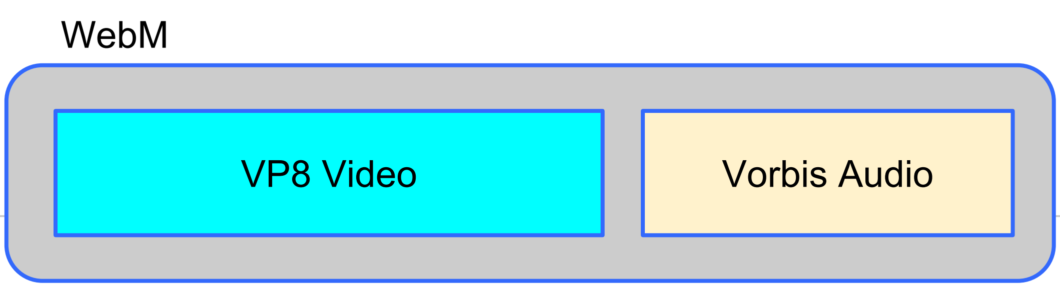
Outline

- Introduction
- VP9 Bit-stream overview
  - Coding Tools
  - Bitstream Features
- Coding results
- Conclusion

Introduction:
The WebM Project

- The Goal of the WebM project:
  - Develop high-quality, *open* video formats for the web, that are freely available to all.
    - Google is dedicated to the *open* web platform, leading to faster innovation, better user experience
    - Video content comprises such a large portion of all web traffic, it must be free as well

- WebM project initially launched in May 2010:
  - VP8 Video+Vorbis Audio+Matroska-like Container



Introduction:
From VP8 to VP9

- Why another codec?
  - Phenomenal growth of online video consumption over the last few years: ***Netflix, YouTube***
    - Majority consumer Internet traffic today is video. Projections indicate the growth will accelerate.
    - Bandwidth is the major cost for providers
  - Consumer expectations of video quality and resolution are also growing:
    - HD is the new default - Ultra HD coming soon
  - Consumers consume video from a variety of power-constrained devices.

- Need a next generation bit-stream that is:
  - more compact, easy to decode, and open (*free*)

Introduction:
VP9 development

- VP9 is the latest open video codec released as part of the WebM project
- Development process:
    - An *experimental* branch at WebM project launch.
    - VP9 development started in earnest late in 2011.
    - Started with re-use of basic building blocks of VP8, but everything was up for change.
    - All development was in the open public *experimental* branch since middle of 2012.
    - Noisy, haphazard process - unlike MPEG
- Released in June 2013
    - [subject to bug-fixes].

Introduction:
Testing Framework

- Typical codec development process:
  - Decide on a reasonable test set
  - Iteratively decide coding tools & parameters based on performance on this test set
  - Caveats:
    - Over-fitting on test set is inevitable. Test set needs to be big enough to ensure sufficient generality
    - Computation is a big problem!
    - Solution: Run encodes in the cloud; cuts down development time significantly.

- VP9 test set during development:
  - *derf*, *std-hd*, *yt*, *yt-hd*: About 100 videos overall

Outline

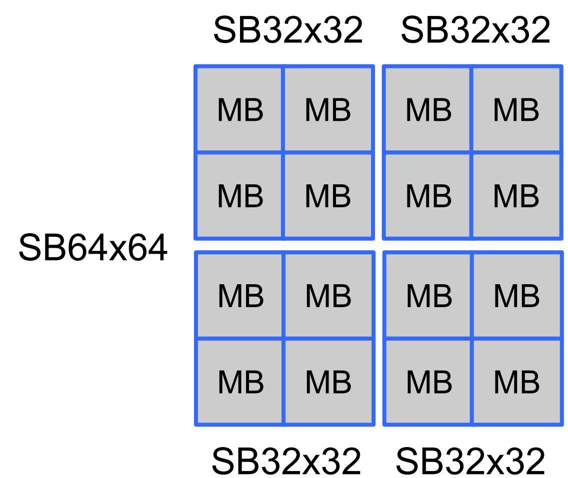
- Introduction
- VP9 Bit-stream overview
    - Coding Tools
    - Bitstream Features
- Coding results
- Conclusion

VP9 Bit-stream Overview:
Coding Tools

- Prediction Block-sizes
- Prediction Modes
  - INTRA Modes
  - INTER References
  - INTER Modes
- Sub-pel Interpolation
- Transforms
- Entropy Coding
- Loop filtering
- Segmentation

Coding Tools:
Prediction Block-sizes

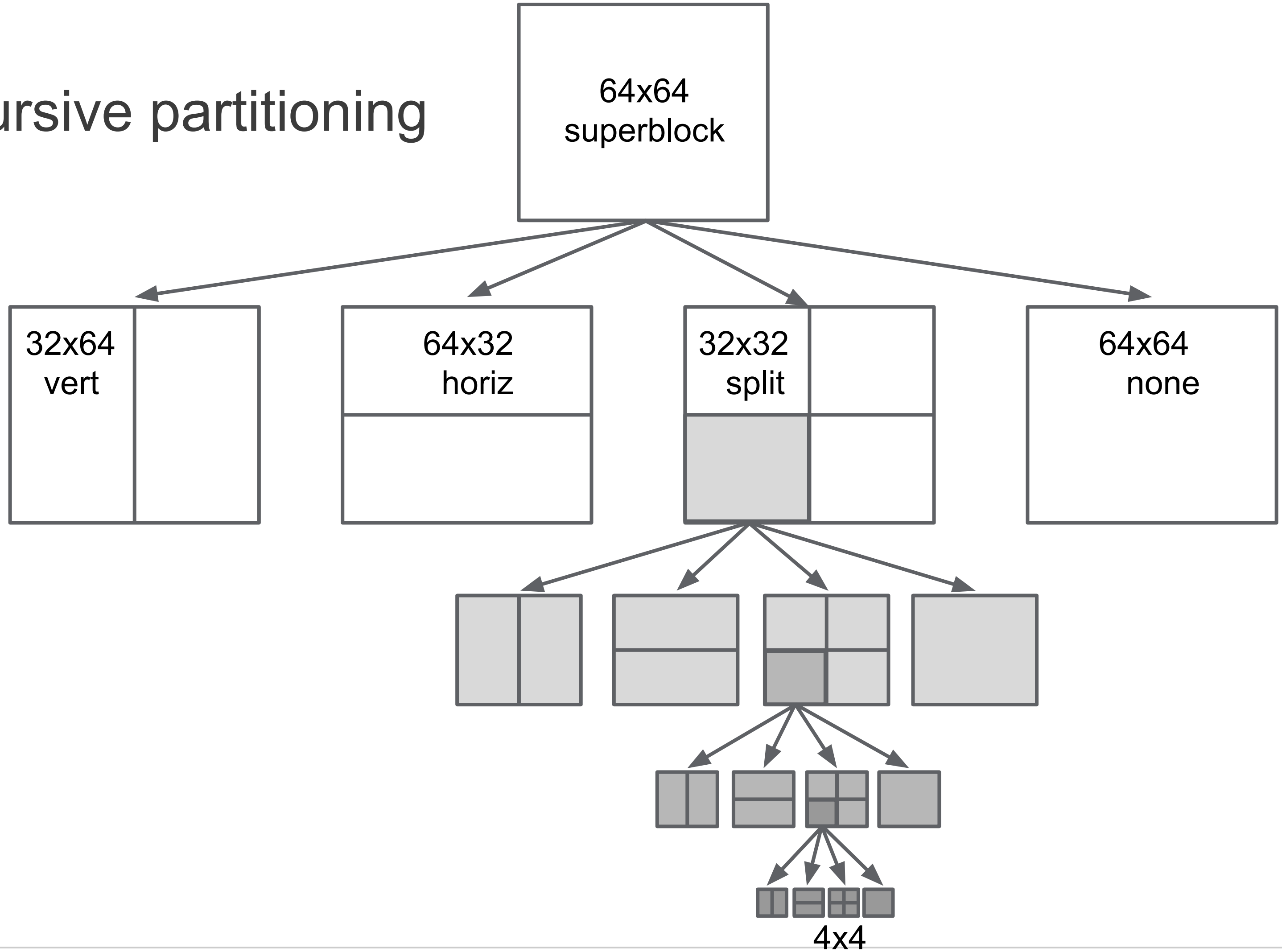
- Generational Progression: HD video material often shows correlation over larger areas
- VP9 introduces Superblocks (SB*mxn*):
  - SB64x64: 64x64 block
  - SB64x32: 64x32 block
  - ...
  - MB16x16: 16x16 block
  - ...
  - SB8x8: 8x8 block



- Mode, ref frame, MV, transform etc. would be grouped together and specified at SB level.
  - Exploits temporal coherence better

Coding Tools:
Prediction Block-sizes

Recursive partitioning

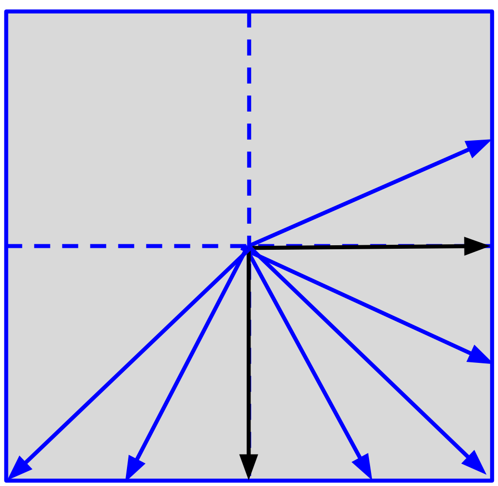


Coding Tools:
Prediction Block-sizes

- Information conveyed at block end-points:
  - Prediction mode (can be INTER or INTRA)
  - If INTER mode:
    - Prediction reference frame(s)
    - Motion vector(s) if needed
    - Sub-pel filter choice
  - Skip flag indicating if there is any non-zero coefficient for prediction residual
  - Transform size
  - Segment index
- For blocks smaller than 8x8, only prediction mode and motions vectors (if needed) are conveyed.

Coding Tools:
Prediction Modes: INTRA Modes

- VP9 uses a total of 10 INTRA predictors
  - 8 directions + DC_PRED and TM_PRED
  - Intra prediction at scales: 4x4, 8x8, 16x16, 32x32 determined by transform size
    - Recursive application of intra prediction followed by reconstruction at the transform size specified



8 directions, along with DC_PRED and TM_PRED modes

Prediction Modes: INTER References

- VP9 allows coding an INTER frame with 3 reference frames
  - Frame header chooses the three reference frames from a pool of 8, as well as specifies the frame buffer (s) the coded frame will replace.
  - Certain frames can be designated *invisible* (Altref)
- For each INTER coded block, either:
  - Use one inter predictor (MV, Ref) - *Single* prediction
  - Combine two single inter predictors by averaging (MV1, Ref1, MV2, Ref2) - *Compound* prediction
    - Ref1 and Ref2 must be different
  - The reference(s) - 1 or 2 - are conveyed at the block end-point

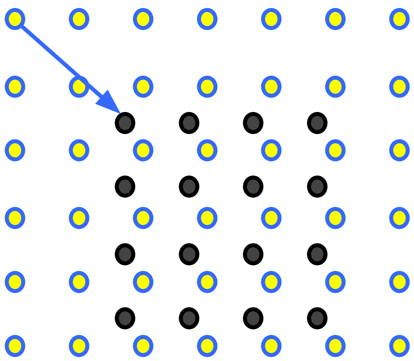
Coding Tools:
Prediction Modes: INTER Modes

- Inter Prediction mode specified per block end-point:
  - NEARESTMV,
  - NEARMV,
  - ZEROMV,
  - NEWMV
- NEARESTMV and NEARMV are the most and second most likely motion vectors for the current block obtained by a survey of MVs in the context for a given reference:
  - Causal neighborhood in current frame
  - Co-located MVs in the previous frame
- In NEWMV mode, NEARESTMV is also used as the motion vector reference
- In *compound* prediction mode, still a single mode is used

Coding Tools:
Sub-pel Interpolation

- Fractional motion critical for video coding performance
  - VP9 supports ⅛th pel motion (1/16 th in U and V)
  - Frame level flag indicates whether ⅛ is to be used
  - Companded motion - use ⅛ pel only for small motion as indicated by reference MV magnitude
- Interpolation filters
  - 3 different 8-tap filters + bilinear
  - 8-tap filters:
    - Regular, Sharp, Smooth
  - Selectable at block or frame level

Reference buffer post loop-filter



Sub-pixel Interpolation Filtering

Coding Tools:
Transforms

Google

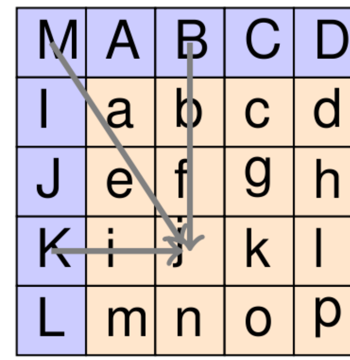
- VP9 uses different transforms for different modes:
  - 2D DCT for INTER modes and
  - Hybrid DCT/ADST transforms for INTRA modes
  - A lossless 4x4 transform for lossless encoding

- A total of 14 all square transforms are used in VP9
  - 4x4:
    - (DCT, DCT), (DCT, ADST), (ADST, DCT), (ADST, ADST)
    - (WHT, WHT) - for lossless mode only
  - 8x8:
    - (DCT, DCT), (DCT, ADST), (ADST, DCT), (ADST, ADST)
  - 16x16:
    - (DCT, DCT), (DCT, ADST), (ADST, DCT), (ADST, ADST)
  - 32x32:
    - (DCT, DCT)

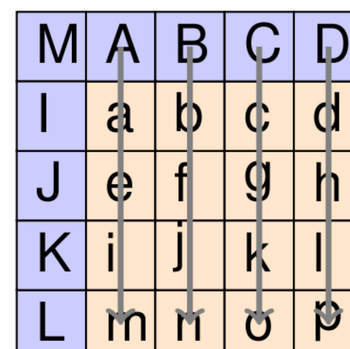
Coding Tools:
Transforms: Asymmetric DST for INTRA

Google

- ADST usage in VP9
    - ADST basis optimal for intra prediction [Han, et. al. 2010] with one-sided boundary
    - Hybrid ADST/DCT dependent on prediction mode for INTRA coding
    - Butterfly ADST - is a special variant amenable to butterfly implementation
        - DST-IV



- True Motion Mode / Diagonal Modes
    - 2-D prediction
    - ADST in vertical direction
    - ADST in horizontal direction
- Vertical Mode / Vertical-Biased Mode
    - 1-D prediction using top boundary
    - ADST in vertical direction
    - DCT in horizontal direction



Coding Tools:
Transforms: Size Selection

- VP9 allows transform size to be smaller than prediction block size, but not larger.
  - Frame level flag indicates one of:
    - a max transform size over a frame
      - All blocks use the max possible transform size up to the max specified
    - or that transform size will be selected for every block end-point

- Special case for INTRA
  - Prediction, transform, quantization, reconstruction is conducted recursively over all blocks of the given transform size

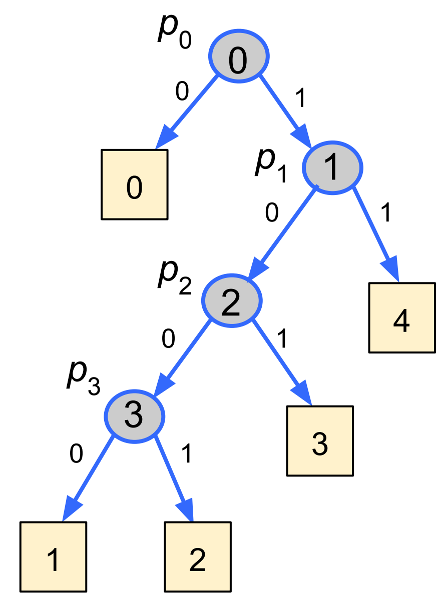
Coding Tools:
Entropy Coding Adaptation

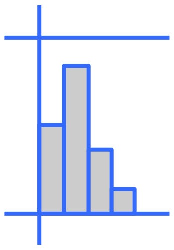
- The VP9 bitstream is fully arithmetic encoded.

- Symbols from an n-ary alphabet
  - Represented as a binary tree:
    - (n - 1) parent nodes,
    - (n) leaf nodes
  - A binary arithmetic coder operates on each parent node
    - 8-bit precision for probabilities

Example:

- A n-ary symbol coded using a tree with (n - 1) binary arithmetic encoder



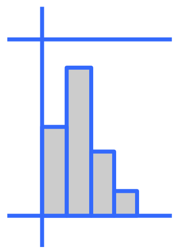


Coding Context

Coding Tools:
Entropy Coding: Forward Adaptation

Google

- Adaptation of probabilities to account for changing statistics is critical in any codec
  - Do not use symbol-by-symbol adaptive entropy coding

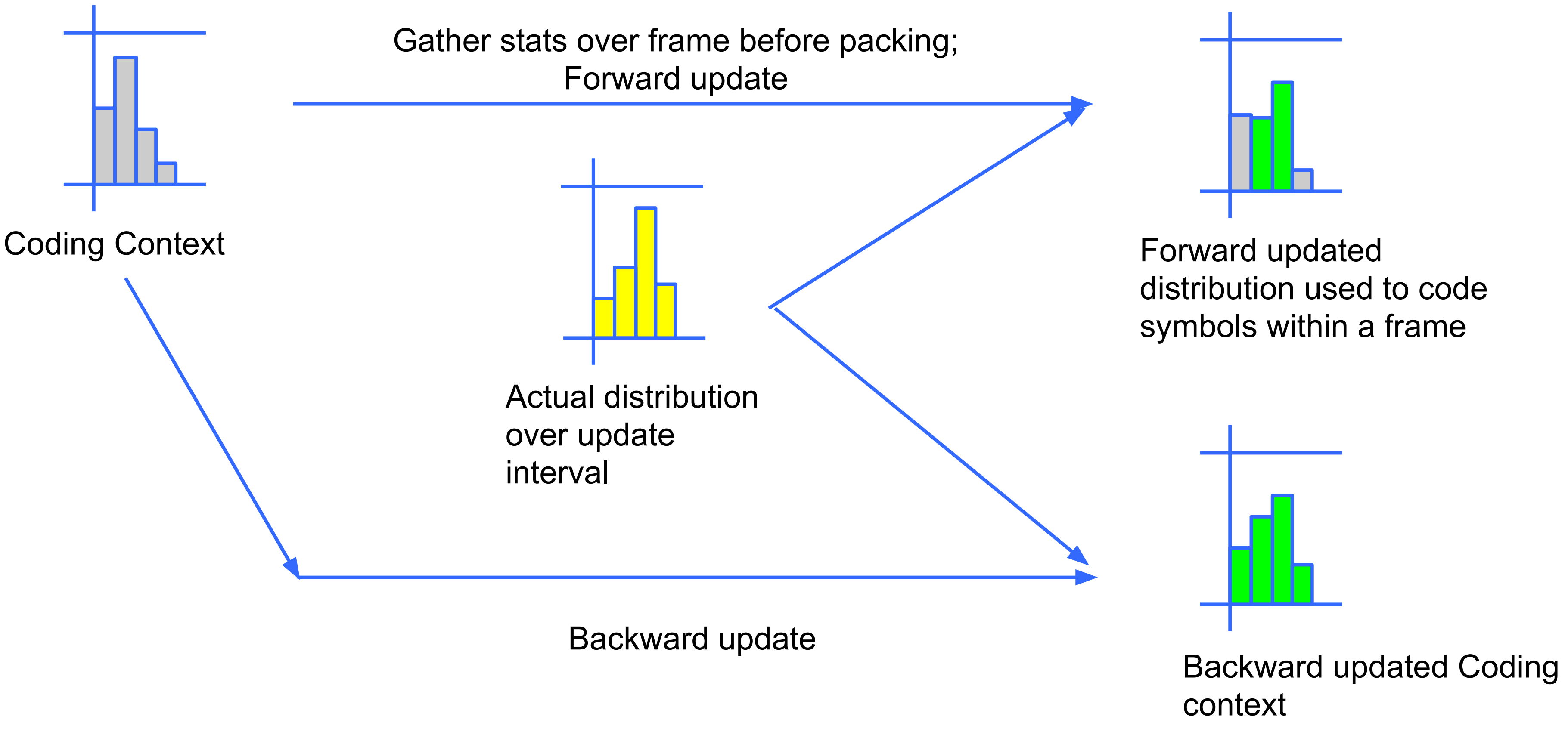


Coding Context

- VP9 combines forward/backward adaptation:
  - Forward adaptation
    - Differentially encoded - sparse updates
      - expensive to conduct a full update
  - Backward adaptation
    - Adaptation only at checkpoints - end of frame
      - Move partially from old probabilities to the new post decode probabilities

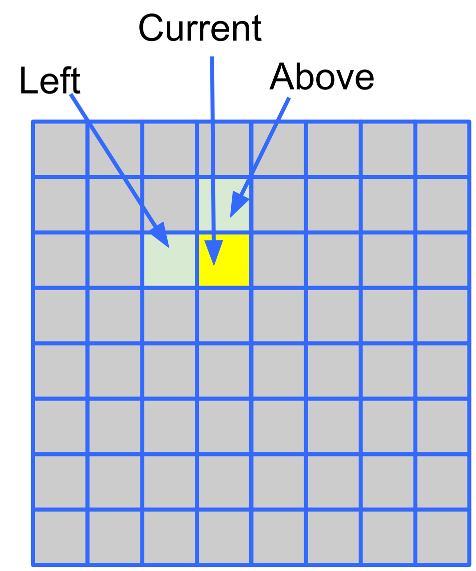
Coding Tools:
Entropy Coding: Forward/Backward

- VP9 uses both forward and backward adaptation:

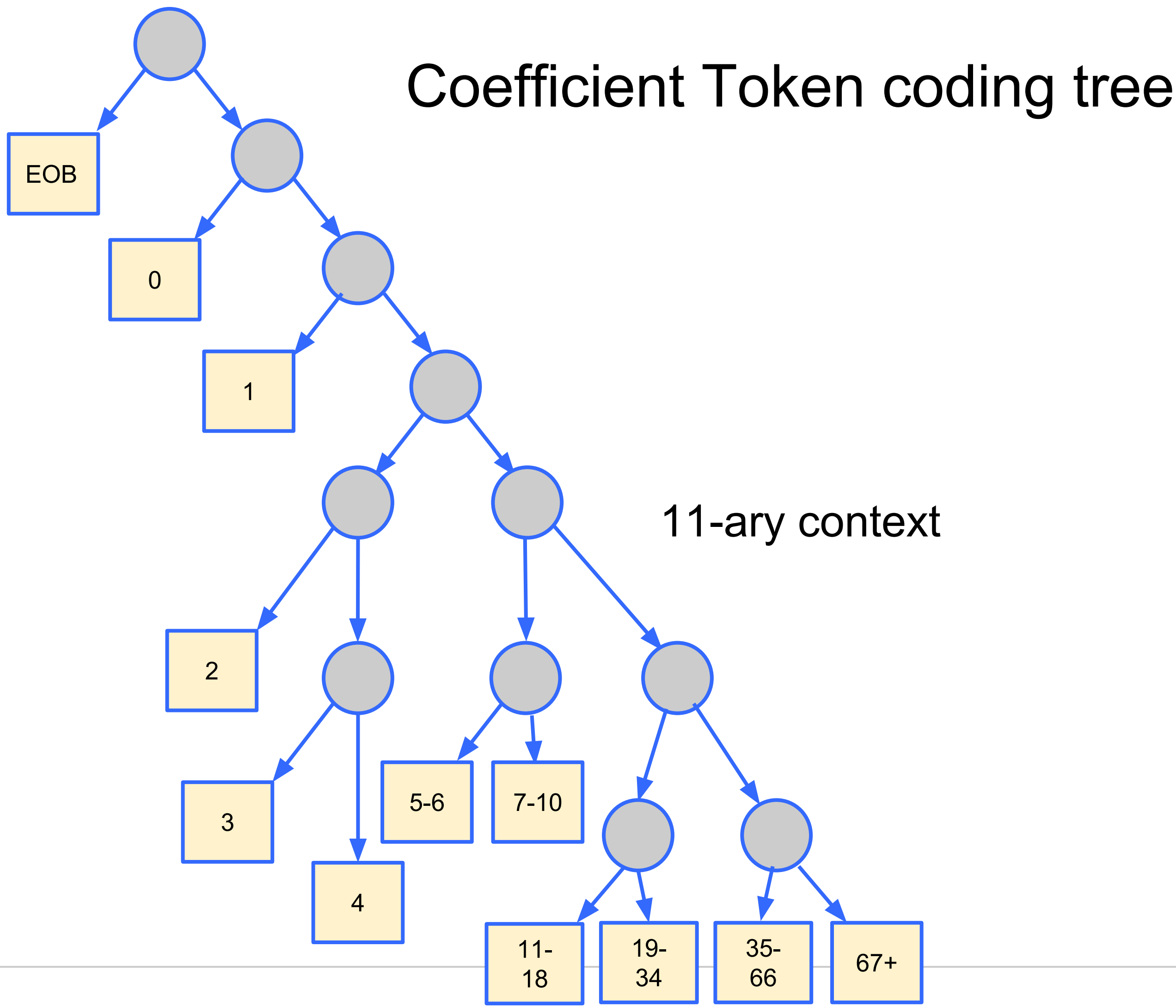


Coding Tools:
Entropy Coding: Coefficient coding

- Coefficient coding
  - Coefficients scanned in a predefined order for given block size
  - Tokens coded:
    - EOB, 0, 1, 2, 3, 4, 5-6, 7-10, 11-18, 19-34, 35-66, 67+
  - Context for coding tokens (576)
    - TX size (4)
    - Y or UV (2)
    - INTRA or INTER (2)
    - BAND (6) - Prior based on coef position in block
    - PREV context (6) - function of above/left coefs already encoded



Coding Tools:
Entropy Coding: Coefficient coding



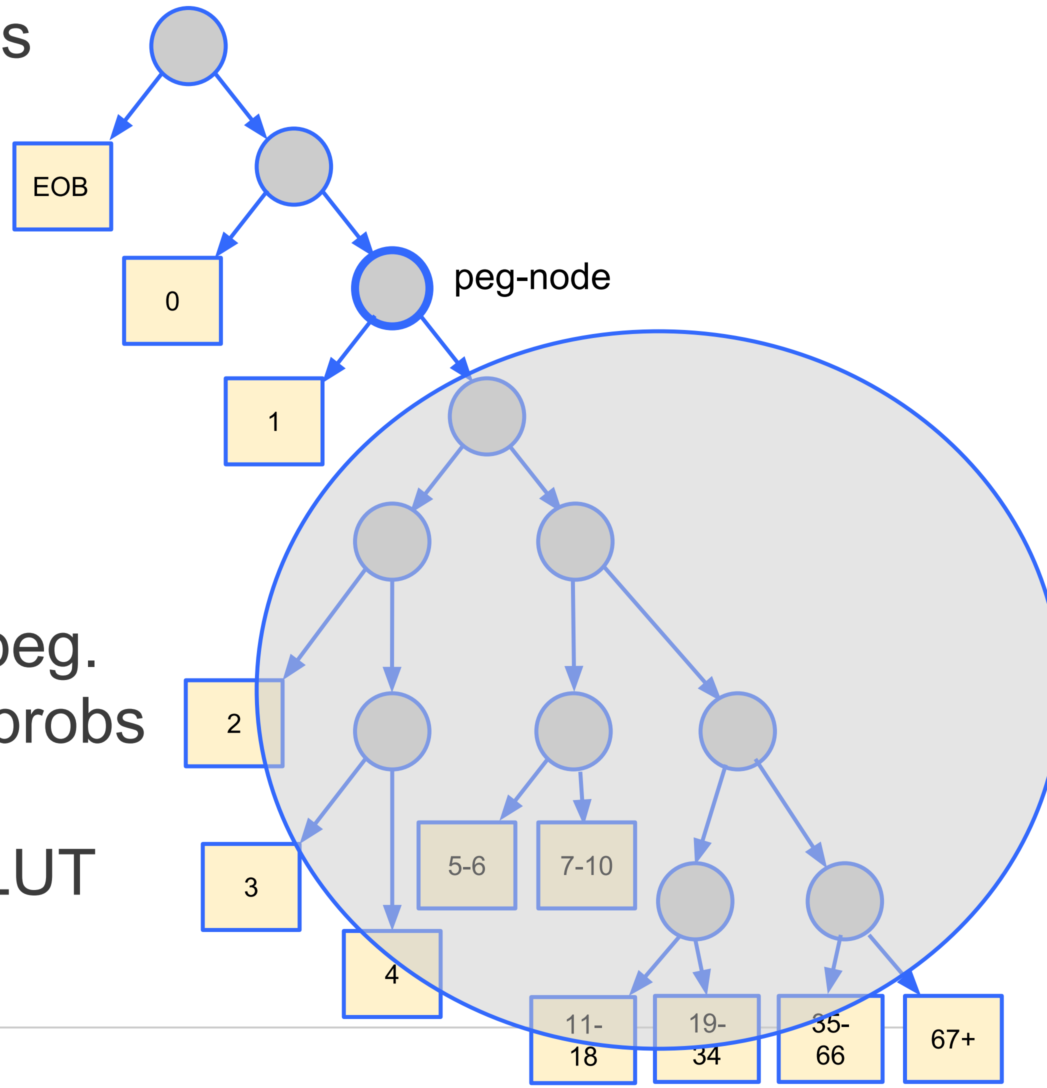
Coding Tools:
Entropy Coding: Modeled Updates

- Maintaining/updating counts for so many (576) contexts is quite complex for decoder - Need a simpler way!

- Modeled Update Approach:
  - Model the coefficients using an appropriate parametric distribution
    - *Symmetric Pareto distribution (power = 8)*
  - Use the probability of one node as a peg to obtain model parameter; then derive the other node probabilities
  - Represent as a look-up table indexed by the peg-node probability
    - Need to maintain counts for only a few nodes

Coding Tools:
Entropy Coding: Modeled updates

- VP9 - modeled updates for both forward and backward updates
  - Update probs of the top 3 nodes only based on real statistics
  - The 1-node is the peg. Derive other node probs based on it
    - Pre-computed LUT



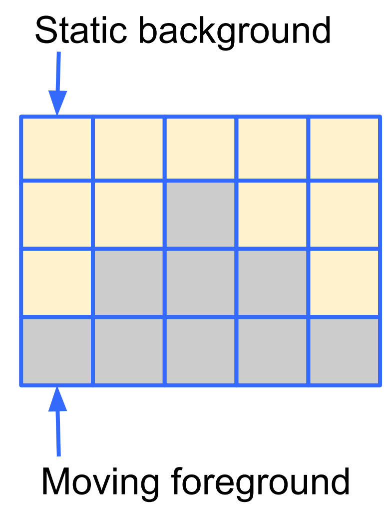
Coding Tools:
Loop Filter

- Designed to reduce blocking
  - VP9 needs to cater to different prediction block-sizes and transform sizes as well as ADST
  - Use filtering across transform block boundaries

- Overall three different filters can used depending on a *flatness test* and transform size.
  - 15-tap: *large txfm + flat*
  - 7-tap: *large txfm + non-flat, medium txfm + flat*
  - 4-point thresholded blur: *medium txfm + non-flat, small txfm*

Coding Tools:
Segmentation

- Segmentation feature significantly enhanced in VP9
    - Groups together blocks that share common characteristics into segments.
    - Indicate segmentation id at block level
        - Differentially encode segmentation map temporally
    - Encode control flags/features at segment level.
        - Q, loop filter strength, ref frame, skip mode



- Unlocking the true potential requires a smart encoder
    - Syntax provides a framework for encoding innovation
    - Various psychovisual optimizations possible

VP9 Bit-stream Overview:
Bit-stream Features

- Error Resilience
- Parallelism
- Scalability

Bitstream Features:
Error-resilience

- VP9 bitstream is arithmetic encoded
  - Errors in bits in a frame will make it impossible to decode subsequent frames
- Error-resilient mode:
  - Allows entropy decoding for successive frames to continue correctly
  - Manage drift until corrective action taken.
- Implementation:
  - Disables features that make entropy decoding across frames dependent on each other
    - Reset coding contexts at every frame
    - MV reference cannot use previous frame MVs
    - Temporal update of segmentation map disabled

Bitstream Features: **Parallelism**

- Critical for smooth (U)HD playback using multi-threaded encode/decode apps on today's multi-core architectures
- Frame-parallel mode:
  - Allows successive frames to be decoded in a quasi-parallel manner
    - Frame header decode *sequential* -> entropy decode *parallel* -> reconstruction *sequential*
- Tiling mode:
  - Allows tiles within the same frame to be decoded in a quasi-parallel manner
    - Entropy decode and reconstruction *parallel* -> loop filtering & entropy backward adaptation *sequential*
- These features are supported in the bitstream, but need development of suitable encode/decode apps.

Bitstream Features:
Scalability

- Spatial Resampling capability:
  - VP9 bit-stream has mid-stream spatial resampling capability
    - Reference buffers and MVs are rescaled
- Types of Scalability
  - Temporal scalability:
    - Reference frame selection ability allows for temporal scalability naturally
  - Spatial scalability:
    - Spatial resampling capability along with reference frame selection enables spatial scalability without explicitly introducing spatial scalable encoding modes.
  - Spatio-temporal scalability:
    - Combine temporal and spatial scalability

Outline

- Introduction
- VP9 Bit-stream overview
  - Coding Tools
  - Bitstream Features
- Coding results
- Conclusion

Coding Results

- Test Codecs
  - VP9 - (Jan 13 head of libvpx repo)
  - H.264/AVC - (X.264 implementation April 2012)
  - HEVC Main - (HM-11.0 implementation)

- Hard to compare codec implementations in a fair way
  - Too many different parameters, too many implementation specific variables
    - Not all implementations support everything
- Difficulty:
  - VP9 only has a 2-pass encoder right now, a basic constant quality mode, but many sophisticated features are not supported yet.
  - HEVC HM11.0 1-pass, gives best PSNR results in fixed Q mode

Coding Results:
Test Conditions

- **Approach** - Fix use cases, then compare codec implementations supporting these use cases
- **Use Cases:**
  - Constant Quality - Achieve a target quality irrespective of bitrate
  - Control Bitrate - Achieve a target bitrate irrespective of quality
- **Key frame intervals**
  - Infinite
  - 5s (typical Youtube) to support seek ability and adaptive streaming
- **Test Sets:**
  - *derf*: 29 Standard CIF sequences
  - *stdhd*: 15 Standard 720P and 1080P sequences
  - *hevchd*: 16 720P and 1080P videos from the HEVC test set
- **Objective Metrics:**
  - BDRATE - % rate delta needed by *test* codec over *baseline* codec at equivalent quality
  - BDSNR/BDSSIM - Quality metric delta achieved by *test* codec over *baseline* codec at equivalent bit-rate

Coding Results:
Constant Quality

- Test 1a: Constant Quality Infinite key frame interval
  - VP9 (constant quality) baseline
    - --end-usage=3 --cpu-used=0 --cq-level=<QP> -kf-max-dist=9999
  - X.264 (crf)
    - --preset veryslow --keyint infinite --tune psnr --profile high --crf <crf-value> --threads 1
  - HEVC (constant QP)
    - -c encoder_randomaccess_main_HM11.cfg -ip -1 --SearchRange=256 --QP=<QP>

- Test 1b: Constant Quality 5s key frame interval
  - VP9 (constant quality) baseline
    - --end-usage=3 --cpu-used=0 --cq-level=<QP> --kf-max-dist=152
  - X.264 (crf)
    - --preset veryslow --keyint 152 --tune psnr --profile high --crf <crf-value> --threads 1
  - HEVC (constant QP)
    - -c encoder_randomaccess_main_HM11.cfg -ip 152 --SearchRange=256 --QP=<QP>

Coding Results:
Control Bitrate

- Test 2a: Bitrate control Infinite key frame interval
  - VP9 (bitrate control) baseline
    - --end-usage=0 --cpu-used=0 --target-bitrate=<Bitrate> --kf-max-dist=9999
  - X.264 (bitrate control)
    - --preset veryslow --keyint infinte --tune psnr --profile high --pass 1 --bitrate <Bitrate> --threads 1
    - --preset veryslow --keyint infinte --tune psnr --profile high --pass 2 --bitrate <Bitrate> --threads 1
  - HEVC (bitrate control)
    - -c encoder_randomaccess_main_HM11.cfg -ip -1 --SearchRange=256 --RateControl=1 --TargetBitrate=<Bitrate>
- Test 2b: Bitrate control 5s key frame interval
  - VP9 (bitrate control) baseline
    - --end-usage=0 --cpu-used=0 --target-bitrate=<Bitrate> --kf-max-dist=152
  - X.264 (bitrate control)
    - --preset veryslow --keyint 152 --tune psnr --profile high --pass 1 --bitrate <Bitrate> --threads 1
    - --preset veryslow --keyint 152 --tune psnr --profile high --pass 2 --bitrate <Bitrate> --threads 1
  - HEVC (bitrate control)
    - -c encoder_randomaccess_main_HM11.cfg -ip 152 --SearchRange=256 --RateControl=1 --TargetBitrate=<Bitrate>

Coding Results:
Averages over test sets

- Average BDRATE - based on Average PSNR

H.264 vs. VP9	Test 1a [CQ-Inf]	Test 1b [CQ-152]	Test 2a [CB-Inf]	Test 2b [CB-152]
derf [29]	26.68%	20.20%	35.85%	25.70%
std-hd [15]	48.45%	42.24%	57.01%	48.78%
hevc-hd [16]	50.50%	39.15%	70.19%	48.99%

HEVC vs. VP9	Test 1a [CQ-Inf]	Test 1b [CQ-152]	Test 2a [CB-Inf]	Test 2b [CB-152]
derf [29]	1.70%	-3.23%	3.83%	-0.77%
std-hd [15]	-4.42%	-6.83%	-4.24%	-7.03%
hevc-hd [16]	-2.59%	-6.93%	-1.67%	-7.09%

Note: Positive (negative) means VP9 is more (less) efficient than the test codec

Coding Results:
Averages over test sets

- Average BDRATE - based on Average SSIM

H.264 vs. VP9	Test 1a [CQ-Inf]	Test 1b [CQ-152]	Test 2a [CB-Inf]	Test 2b [CB-152]
derf [29]	42.17%	30.49%	55.14%	37.60%
std-hd [15]	71.82%	60.90%	78.29%	63.86%
hevc-hd [16]	75.69%	55.30%	95.50%	62.24%

HEVC vs. VP9	Test 1a [CQ-Inf]	Test 1b [CQ-152]	Test 2a [CB-Inf]	Test 2b [CB-152]
derf [29]	6.41%	-1.20%	11.47%	3.73%
std-hd [15]	1.28%	-2.53%	0.88%	-3.28%
hevc-hd [16]	0.64%	-5.59%	3.73%	-3.40%

Note: Positive (negative) means VP9 is more (less) efficient than the test codec

Coding Results:
Summary

- VP9 - performance
  - Quite competitive with HEVC on diverse test material with very long key frame intervals
    - Modest degradation at lower key frame intervals
  - VP9 generally performs better on SSIM scores than PSNR

- Disclaimer on test conditions
  - Black-box testing comparing X.264, HM11 and libvpx
  - Hard to make apples to apples comparison today
    - 1-pass vs. 2-pass
    - Implementation of encode features such as pyramid B-frames
- Open-questions:
  - What is a good test set representative of Web video today ?
  - How to design a 'fair' test between multiple codecs that compares coding tools rather than implementations ?

Outline

- Introduction
- VP9 Bit-stream overview
  - Coding Tools
  - Bitstream Features
- Coding results
- Conclusion

Conclusion:
The master branch

- Current status of the VP9 project
  - Active work in the master branch of the libvpx repository to increase encode/decode speed, support multiple platforms, use-cases etc.
- Currently only a good 2-pass encoder exists. To come:
  - Better one-pass encoder
  - Better real-time, low-delay encoder
  - Encoders that can exploit bit-stream features - such as segmentation, hierarchical Altref frames
- Contributions welcome!

Conclusion:
The experimental branch

- When VP9 was released in June 2013:
  - The *experimental* branch was merged into the *master* branch of the libvpx repository
- Experimental branch is still alive
  - Intended to be a platform for research and development for a next-generation bitstream
  - Already some new experiments have been added that is 2% better than the master branch
- Invitation to developers and researchers to actively participate in developing new open-source codec technologies
  - Comprehensive testing framework in google cloud
  - A new way of developing video codecs

Q & A