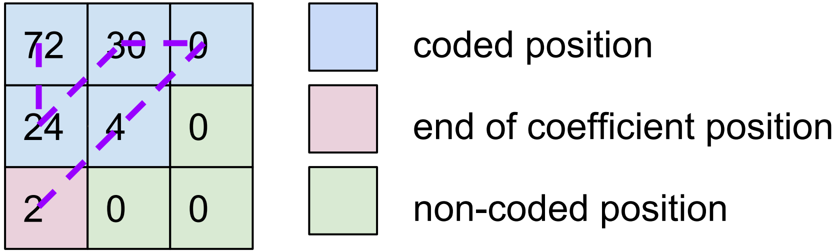
A CONSTRAINED ADAPTIVE SCAN ORDER APPROACH TO TRANSFORM COEFFICIENT ENTROPY CODING

Ching-Han Chiang, Jingning Han, Yaowu Xu
Google

Coding of Transform Block Coefficients

- Use a scan order to code transform block coefficients sequentially.
- Code an end-of-coefficient token after last non-zero coefficient in the scan order.
- The rest of zero coefficients can be skipped.



72	30	0
24	4	0
2	0	0

🟦 coded position

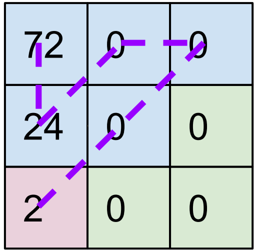
🟪 end of coefficient position

🟩 non-coded position

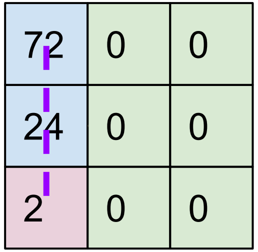
72 -> 24 -> 30 -> 0 ->4 -> 2 -> eob

Motivation

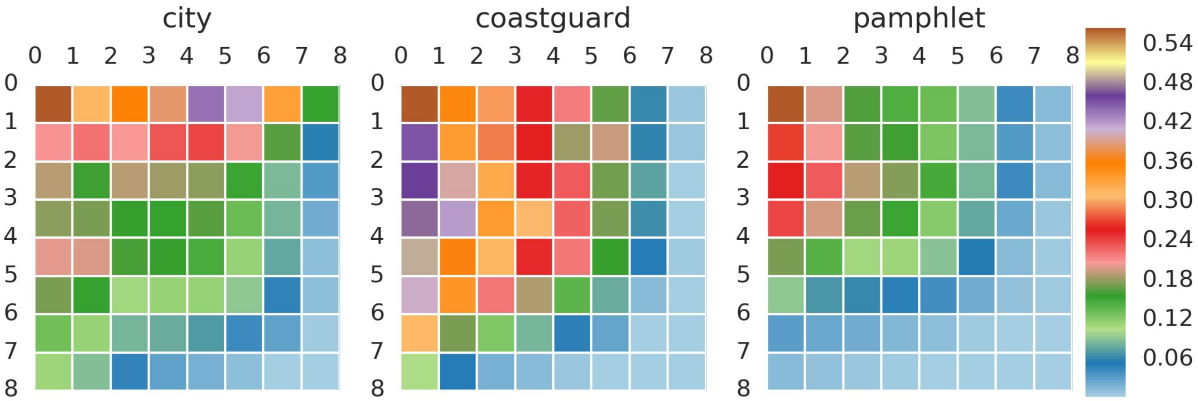
- Arrange most zero coefficients to the tail.
  - The end-of-coefficient token will allow us to ignore zero coefficients in the tail.
- Transform block statistics are different across different video clips.
  - Using a predefined scan order will waste bits in coding zero coefficients.
  - An adaptive approach is needed.



72	0	0
24	0	0
2	0	0

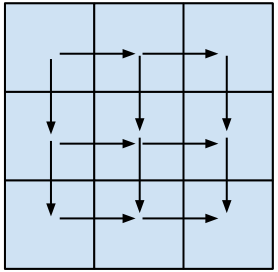


72	0	0
24	0	0
2	0	0



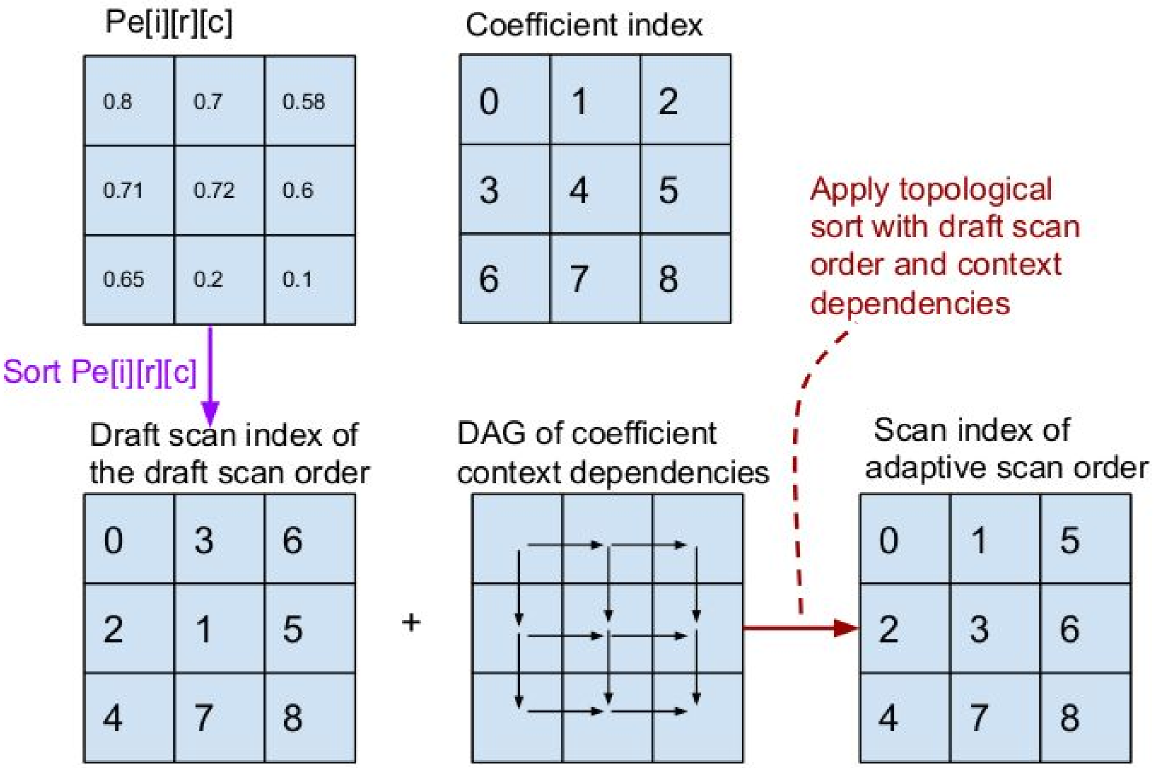
Context Dependency

- How about we design an adaptive scan order merely by sorting the non-zero probabilities of positions in transform blocks?
- The above/left coefficient context dependency is applied in entropy coding to exploit the remaining inter-coefficient correlations.
- The scan order obtained from sorting the non-zero probabilities may have conflict with the context dependency.



Topological Sort - Example

- By applying topological sort, one can generate a scan order that mostly follows the descending order of non-zero probabilities without violating the context dependency.



Pe[i][r][c]

0.8	0.7	0.58
0.71	0.72	0.6
0.65	0.2	0.1

Coefficient index

0	1	2
3	4	5
6	7	8

Sort Pe[i][r][c]

Draft scan index of the draft scan order

0	3	6
2	1	5
4	7	8

+

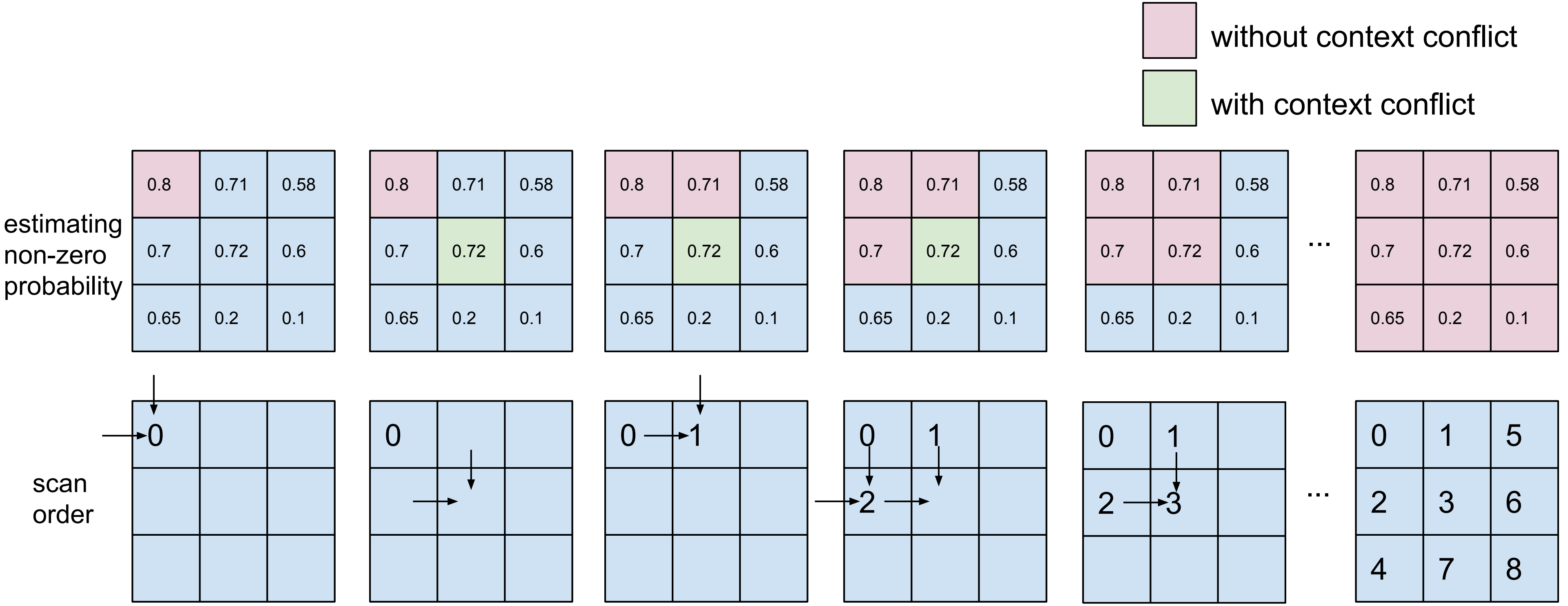
DAG of coefficient context dependencies

Apply topological sort with draft scan order and context dependencies

Scan index of adaptive scan order

0	1	5
2	3	6
4	7	8

Topological Sort - Resolve Context Conflict



🟥 without context conflict

🟩 with context conflict

estimating non-zero probability

0.8	0.71	0.58
0.7	0.72	0.6
0.65	0.2	0.1

0.8	0.71	0.58
0.7	0.72	0.6
0.65	0.2	0.1

0.8	0.71	0.58
0.7	0.72	0.6
0.65	0.2	0.1

0.8	0.71	0.58
0.7	0.72	0.6
0.65	0.2	0.1

0.8	0.71	0.58
0.7	0.72	0.6
0.65	0.2	0.1

...

0.8	0.71	0.58
0.7	0.72	0.6
0.65	0.2	0.1

scan order

0		

0		

0	1	

0	1	
2		

0	1	
2	3	

...

0	1	5
2	3	6
4	7	8

Estimation of Non-zero Probabilities of Transform Coefficients

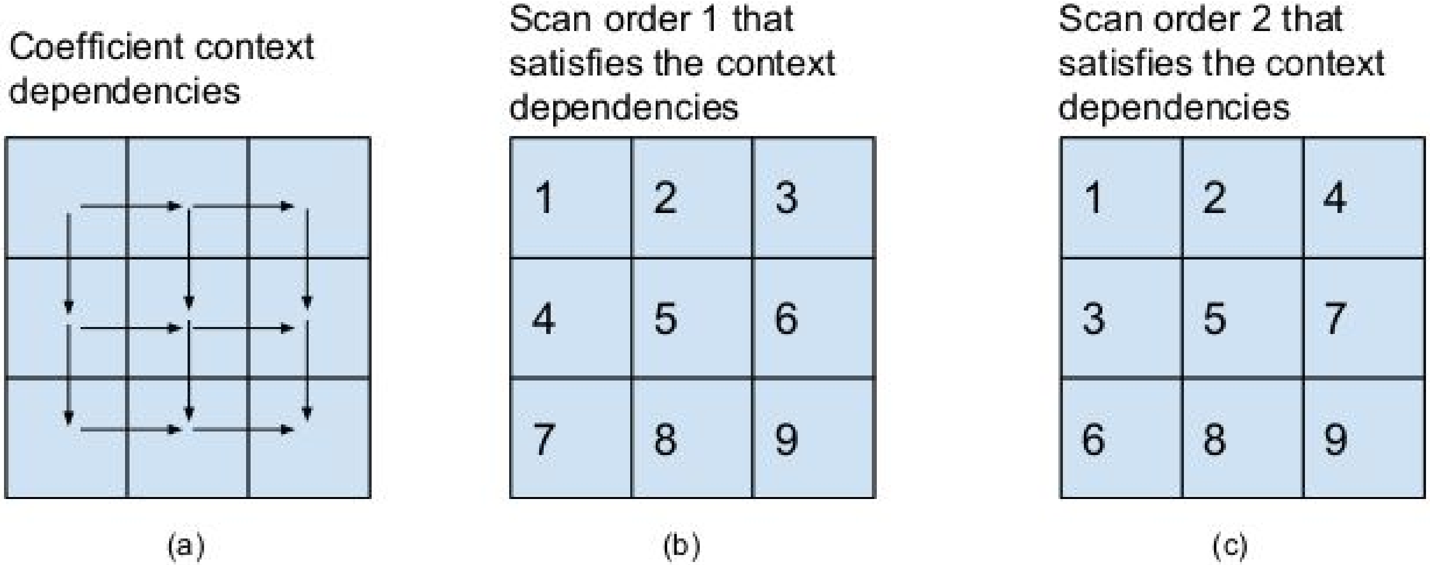
- The cost of transmitting non-zero probabilities or the adaptive scan order from encoder to decoder is impractically large
- Moving window estimation
    - Estimate non-zero probabilities of transform block coefficients for i-th frame.
    - Non-zero coefficient counts and number of transform blocks
        - $C[i-1][r][c]$
        - $M \rightarrow$ number of transform blocks
    - Observed non-zero probabilities
        - $P c [i-1][r][c] = C[i-1][r][c]/M$
        - per-frame update
    - Estimating non-zero probabilities
        - $P e [i][r][c] = (1 - k) * P e [i-1][r][c] + k * P c [i-1][r][c]$
        - per-frame update

Performance

- Coding gains (BDRate) over predefined scan order scheme on VP9
    - Low-resolution dataset
        - 40 videos with resolutions of 240p, SIF or CIF
        - 1.04%
    - Mid-resolution dataset
        - 23 videos with resolutions of 480p or 4CIF
        - 0.93%
    - High-resolution dataset
        - 38 videos with resolutions 720p, 1080p or XGA
        - 1.13%
- Encoder/decoder Time
    - Encoder: non-observable
    - Decoder: +1%

Q & A

Context Dependency - A Sparse but Strict Constraint



Coefficient context dependencies

(a)

Scan order 1 that satisfies the context dependencies

1	2	3
4	5	6
7	8	9

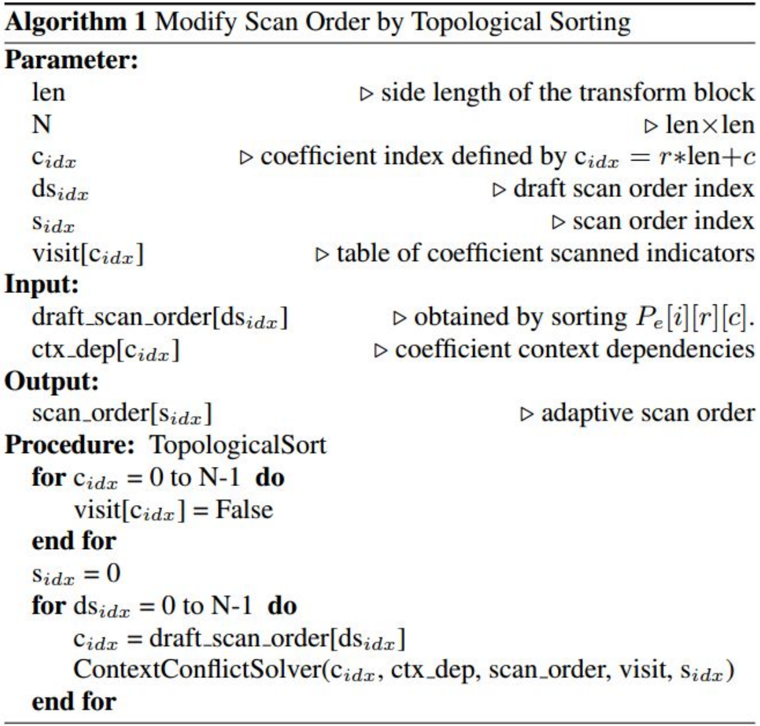
(b)

Scan order 2 that satisfies the context dependencies

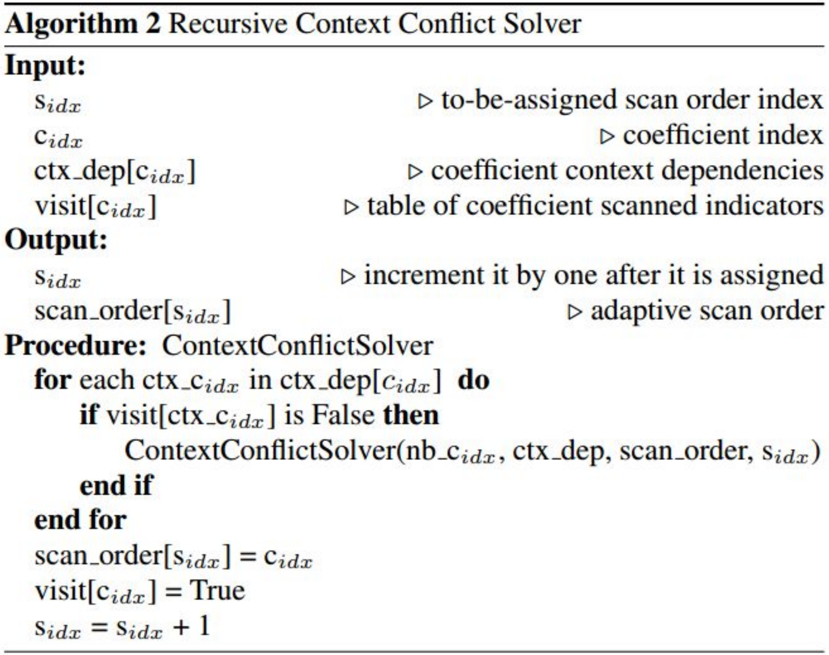
1	2	4
3	5	7
6	8	9

(c)

Topological Sort - Pseudocode



```
Algorithm 1 Modify Scan Order by Topological Sorting
Parameter:
    len                         ▷ side length of the transform block
    N                           ▷ len×len
    $c_{idx}$                   ▷ coefficient index defined by $c_{idx} = r*len+c$
    $ds_{idx}$                  ▷ draft scan order index
    $s_{idx}$                   ▷ scan order index
    visit[$c_{idx}$]            ▷ table of coefficient scanned indicators
Input:
    draft_scan_order[$ds_{idx}$]   ▷ obtained by sorting $P_e[i][r][c]$.
    ctx_dep[$c_{idx}$]             ▷ coefficient context dependencies
Output:
    scan_order[$s_{idx}$]          ▷ adaptive scan order
Procedure: TopologicalSort
    for $c_{idx}$ = 0 to N-1 do
        visit[$c_{idx}$] = False
    end for
    $s_{idx}$ = 0
    for $ds_{idx}$ = 0 to N-1 do
        $c_{idx}$ = draft_scan_order[$ds_{idx}$]
        ContextConflictSolver($c_{idx}$, ctx_dep, scan_order, visit, $s_{idx}$)
    end for
```



```
Algorithm 2 Recursive Context Conflict Solver
Input:
    $s_{idx}$                   ▷ to-be-assigned scan order index
    $c_{idx}$                   ▷ coefficient index
    ctx_dep[$c_{idx}$]          ▷ coefficient context dependencies
    visit[$c_{idx}$]            ▷ table of coefficient scanned indicators
Output:
    $s_{idx}$                   ▷ increment it by one after it is assigned
    scan_order[$s_{idx}$]       ▷ adaptive scan order
Procedure: ContextConflictSolver
    for each ctx_$c_{idx}$ in ctx_dep[$c_{idx}$] do
        if visit[ctx_$c_{idx}$] is False then
            ContextConflictSolver(nb_$c_{idx}$, ctx_dep, scan_order, $s_{idx}$)
        end if
    end for
    scan_order[$s_{idx}$] = $c_{idx}$
    visit[$c_{idx}$] = True
    $s_{idx}$ = $s_{idx}$ + 1
```