Computer Vision Final Project: Global Motion Compensation

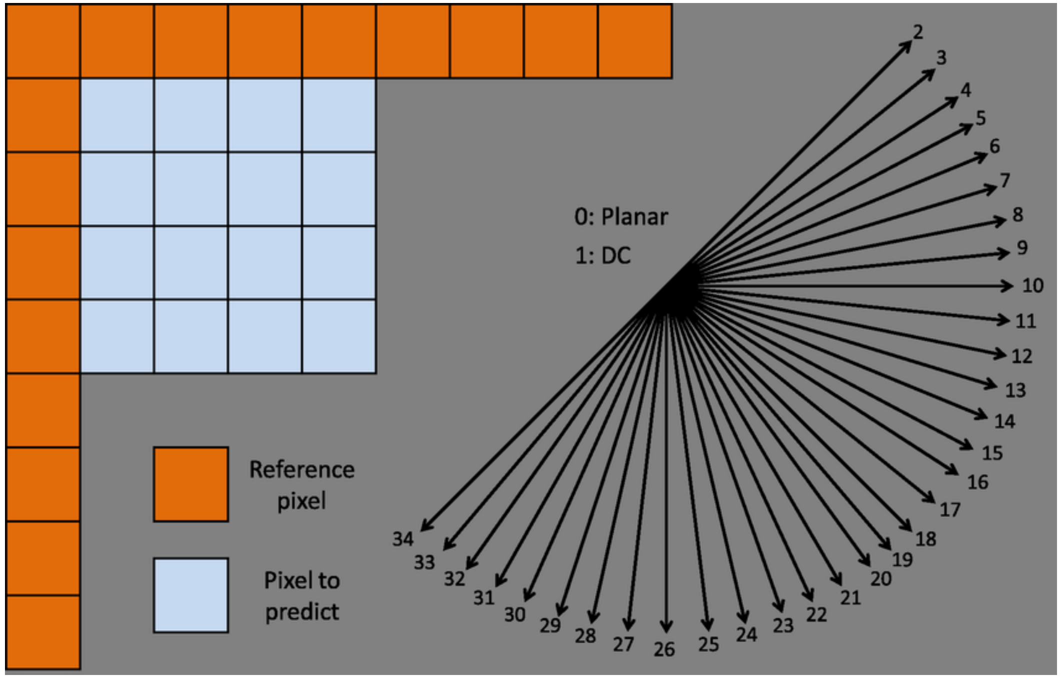
莊子德 peter.chuang@mediatek.com

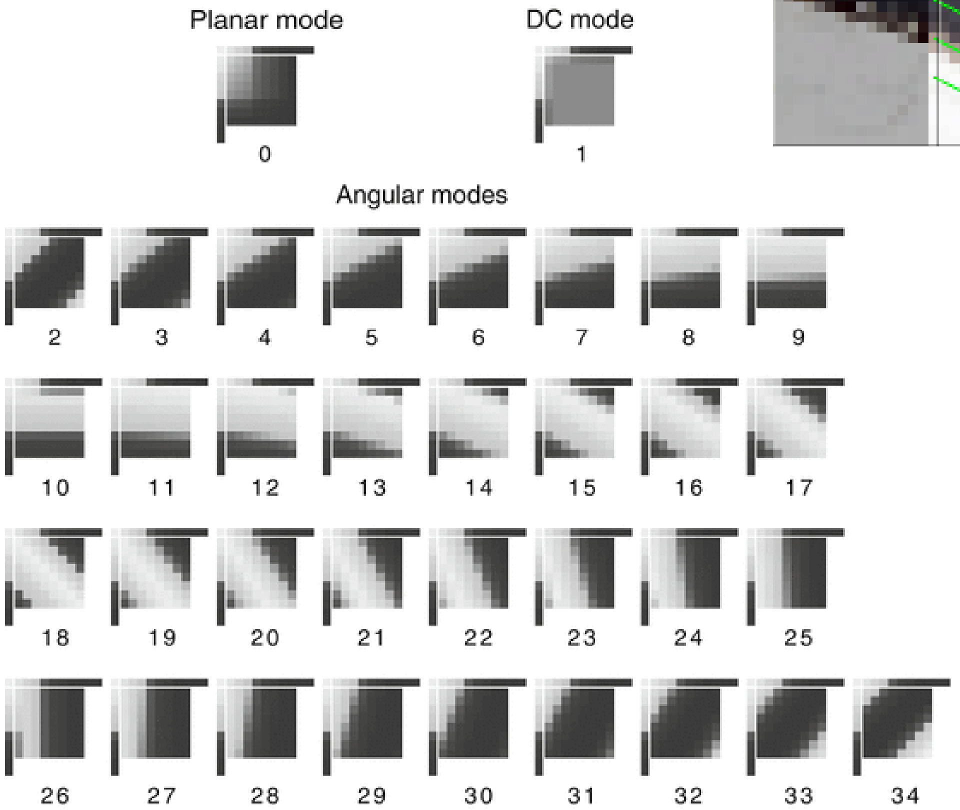
Concept of Video Coding

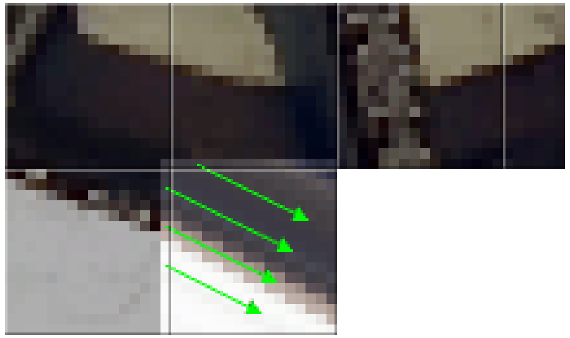
- Video is composed of a series of images
- The raw data of video is too large for storing or transmission
  - The raw data of a 2-hour full HD video requires 1,343 GB (1.5Gbits/sec)
  - After compression, it may only require 9GB (10Mbits/sec)
- Key concept of video coding is to remove redundancy
  - Spatial redundancy
  - Temporal redundancy
  - Statistic redundancy

Intra Prediction to Remove Spatial Redundancy

- Using the neighboring pixels to predict the current block
- Only encode the prediction direction and residual

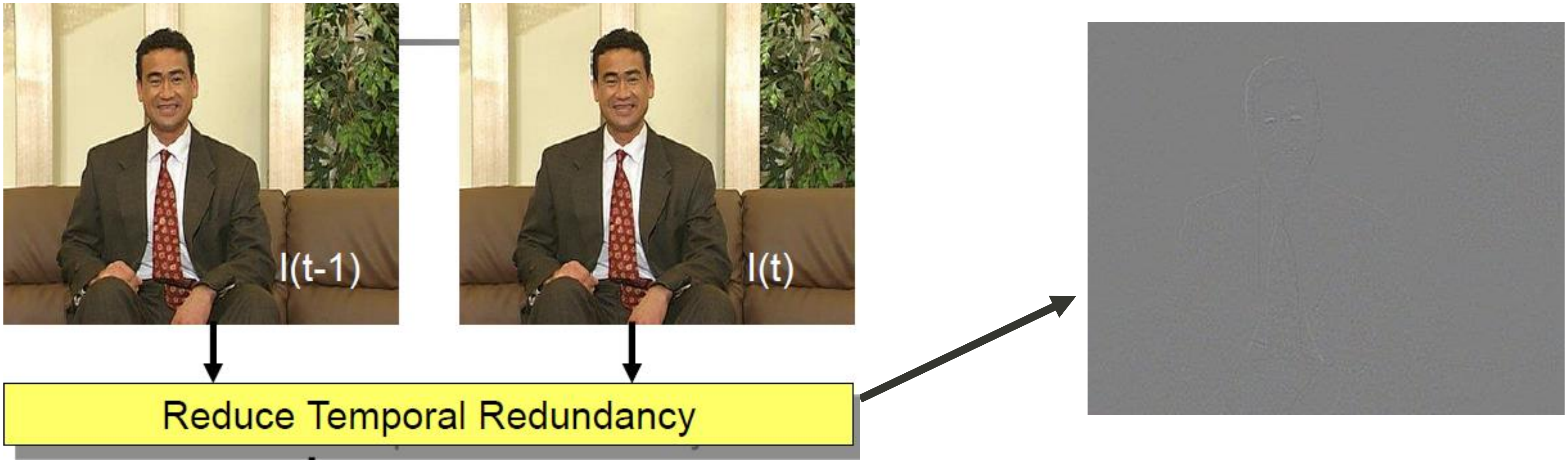






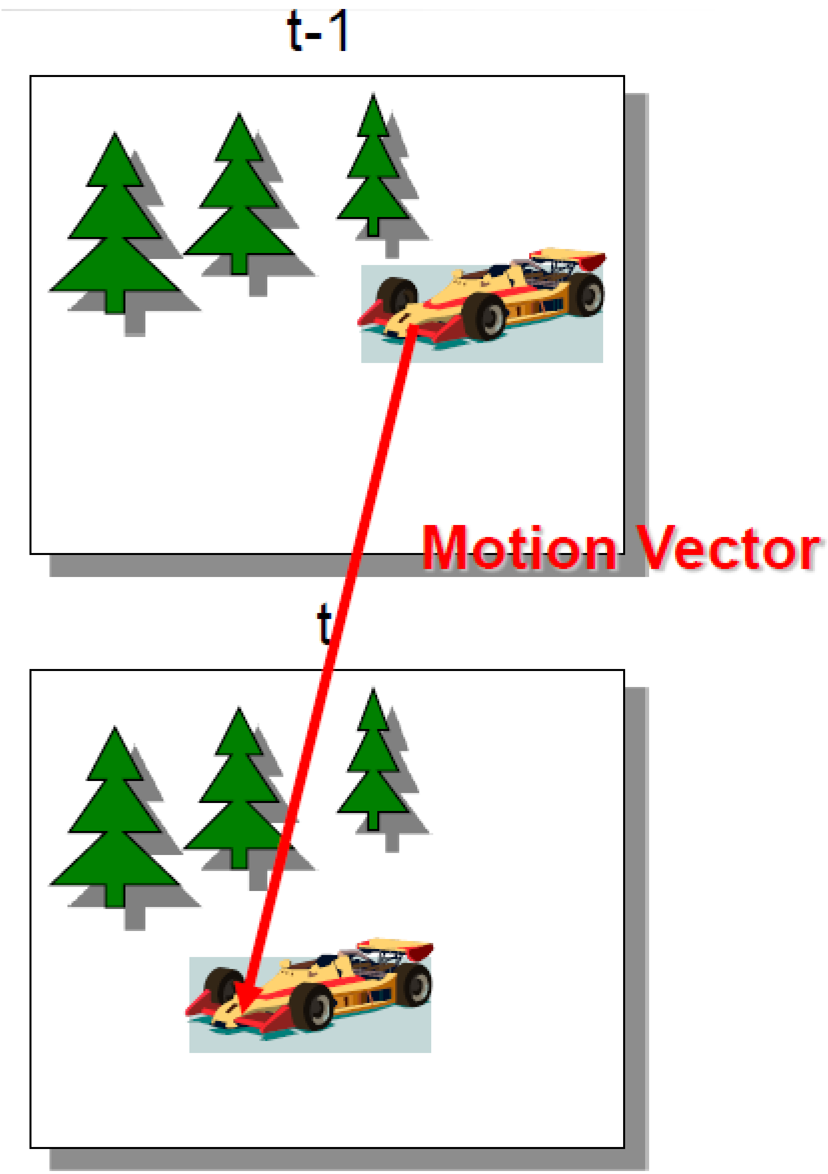
Inter Prediction to Remove Temporal Redundancy

- Using the previous coded picture to predict the current block
- Only encode the motion vector and residual
- Most common inter prediction tool is translational motion estimation and motion compensation



Translational Motion Estimation and Motion Compensation

- **Motion Estimation**
  - To derive the motion information between pictures
  - Find the motion vector (MV) for a current block
- **Motion Compensation**
  - Copy the block of pixels pointed by the MV to the current block
- **Translational motion model only deal with panning and shifting**

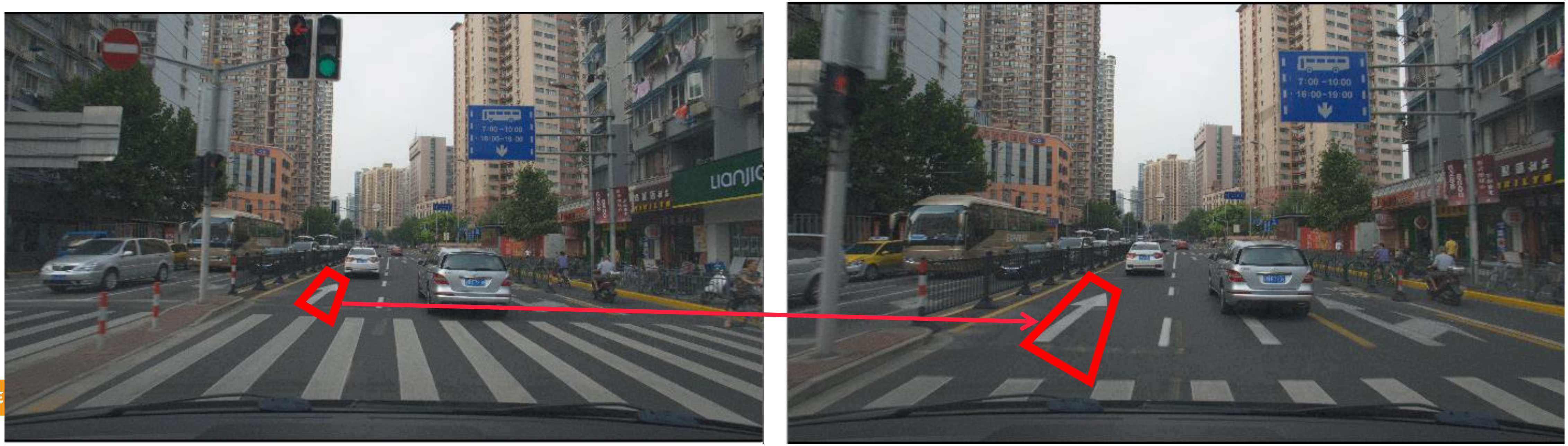


Project Goal

- Motion compensation plays an important role in video coding that can remove temporal redundancy efficiently. Most of video coding standards only support local translation motion model.
- However, in the real world, lots of camara captured videos may need higher degree of motion models (e.g. affine motion model, projection/perspective motion model) for compensating the temporal redundancy.

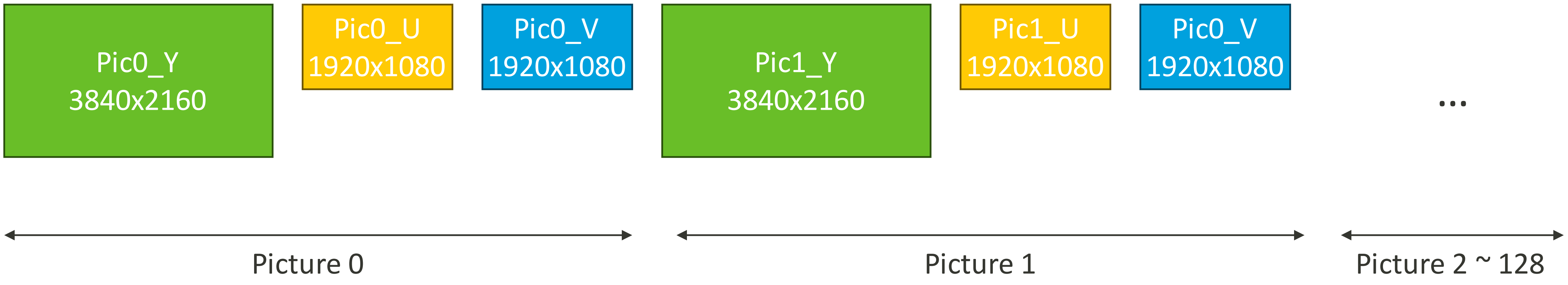
Project Goal

- In this final project, you need to design a global motion compensation (GMC) system to remove the temporal redundancy of a video with global motion



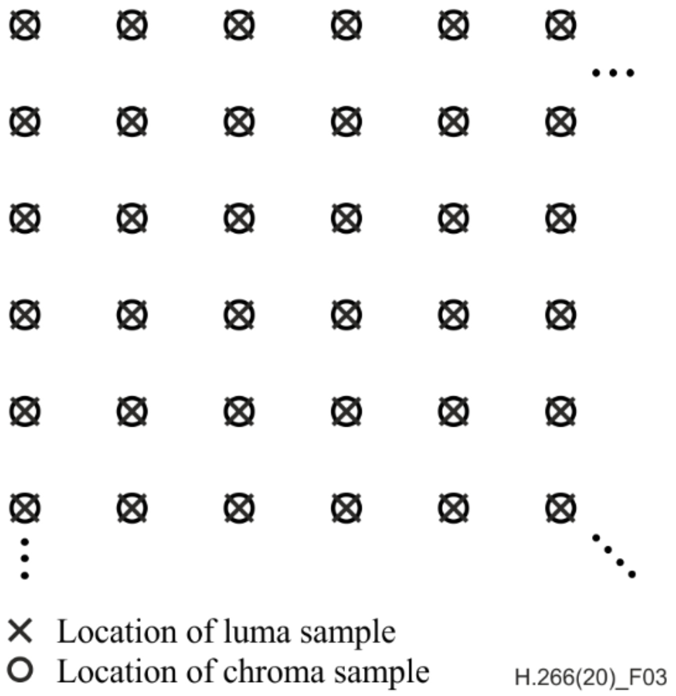
What You Need to Do

1. **Download the target video sequence from TA**
   - The target video sequence is a 129 frames YUV420 8-bits video.
   - The data format of the sequence is in the order of “Pic0_Y -> Pic0_U -> Pic0_V -> Pic1_Y -> Pic1_U -> Pic1_V -> … Pic128_Y -> Pic128_U -> Pic128_V”, where the PicX_Y has 3840x2160 bytes, the PicX_U has 1920x1080 bytes, and the PicX_V has 1920x1080 bytes.

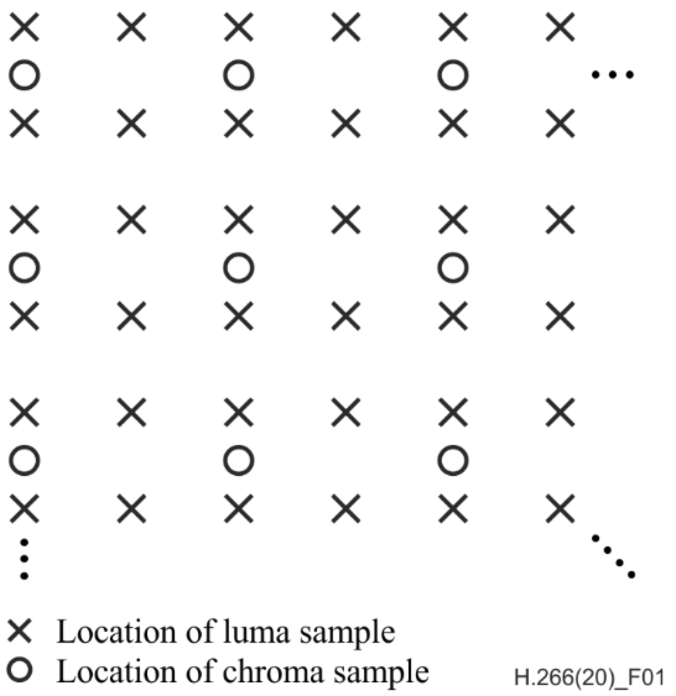


YUV420 Format

- **YUV444 format**
  - One luma sample with one U and one V samples
  - Can be transformed to RGB444 directly



- **YUV420 format**
  - 4 luma samples with one U and one V samples
  - Need upsampling for chroma samples before transformed to RGB444



What You Need to Do

2. **Please process the picture 1-31, 33-63, 65-95, 97-127 (no need to process picture 0, 32, 64, 96, and 128) and follow the Hierarchical-B processing order to process the video.**
   - The available reference pictures are also defined
   - The target picture marked as “X” means that you don’t need to process this picture.

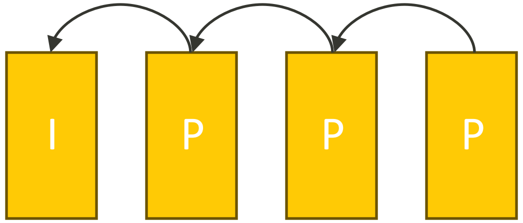
Processing order	Target Picture	Reference Pic0	Reference Pic1
0	0 (X)		
1	32 (X)		
2	16	0	32
3	8	0	16
4	4	0	8
5	2	0	4
6	1	0	2
7	3	2	4
8	6	4	8
9	5	4	5
10	7	6	8
11	12	8	16
12	10	8	12
13	9	8	10
14	11	10	12

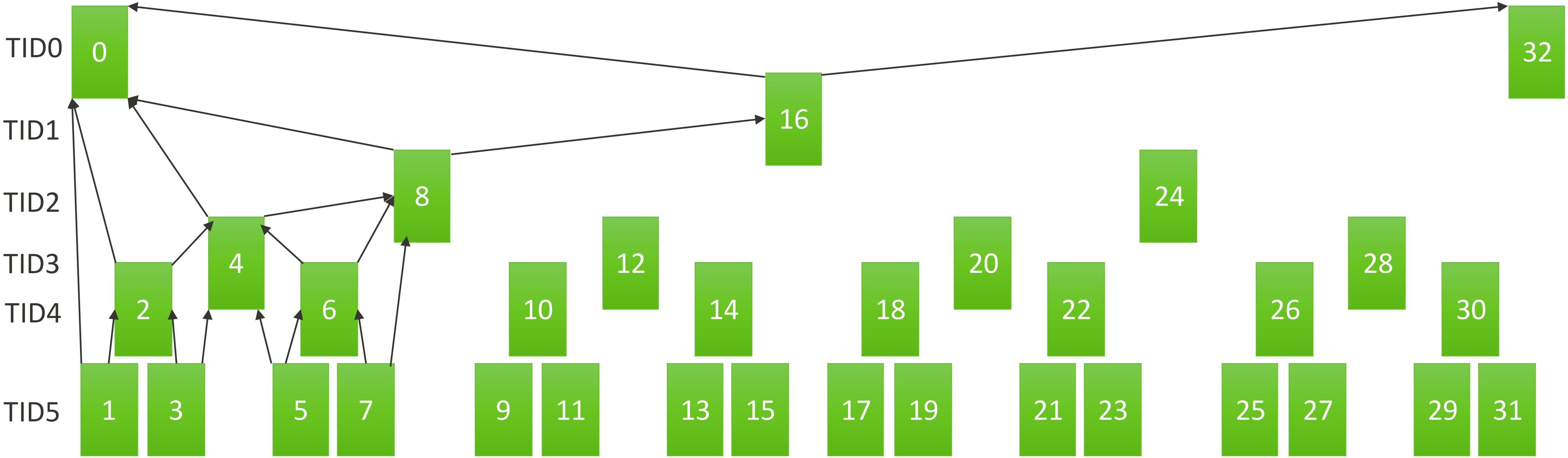
Processing order	Target Picture	Reference Pic0	Reference Pic1
32	31	30	32
33	64 (X)		
34	48	32	64
35	40	32	48
36	36	32	40
37	34	32	36
38	33	32	34
39	35	34	36
40	38	36	40
41	37	36	37
42	39	38	40
43	44	40	48
44	42	40	44
45	41	40	42
46	43	42	44

Processing order	Target Picture	Reference Pic0	Reference Pic1
96	95	94	96
97	128 (X)		
98	112	96	128
99	104	96	112
100	100	96	104
101	98	96	100
102	97	96	98
103	99	98	100
104	102	100	104
105	101	100	101
106	103	102	104
107	108	104	112
108	106	104	108
109	105	104	106
110	107	106	108

Hierarchical-B Coding Structure

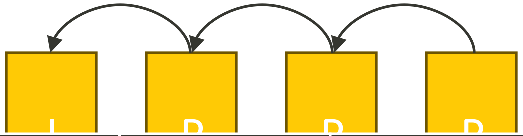
- Can provide better coding efficiency than IPPP
- Can support temporal scalability



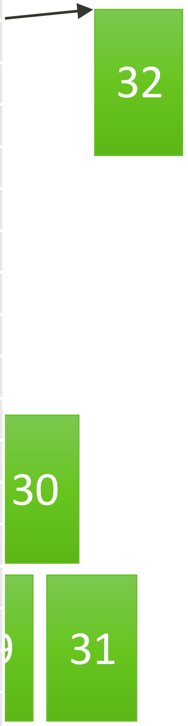


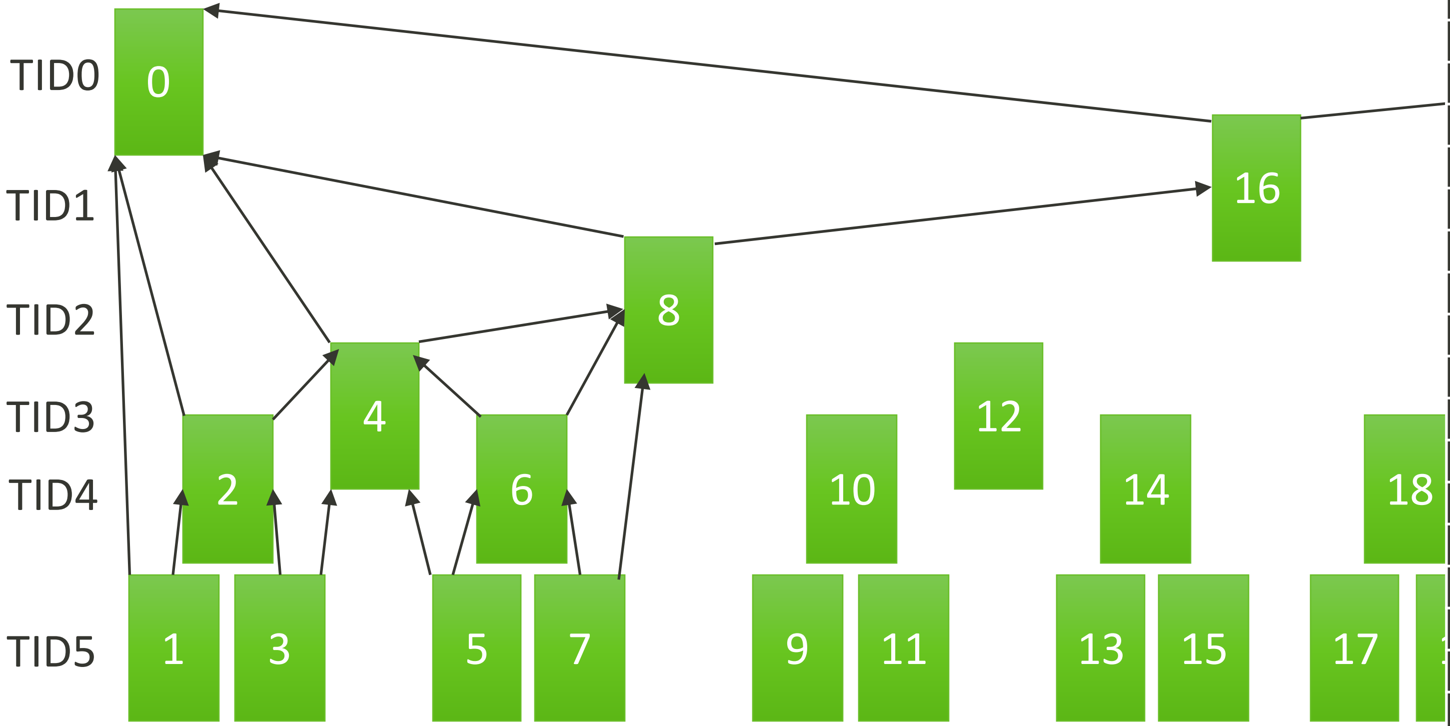
Hierarchical-B Coding Structure

- Can provide better coding efficiency than IPPP
- Can support temporal scalability



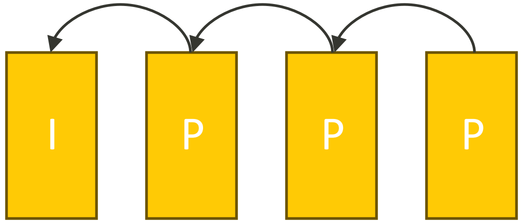
Processing order	Target Picture	Reference Pic0	Reference Pic1
0	0 (X)		
1	32 (X)		
2	16	0	32
3	8	0	16
4	4	0	8
5	2	0	4
6	1	0	2
7	3	2	4
8	6	4	8
9	5	4	5
10	7	6	8
11	12	8	16
12	10	8	12
13	9	8	10
14	11	10	12
15	14	12	16
16	13	12	13
17	15	14	16
18	24	16	32
19	20	16	24
20	18	16	20





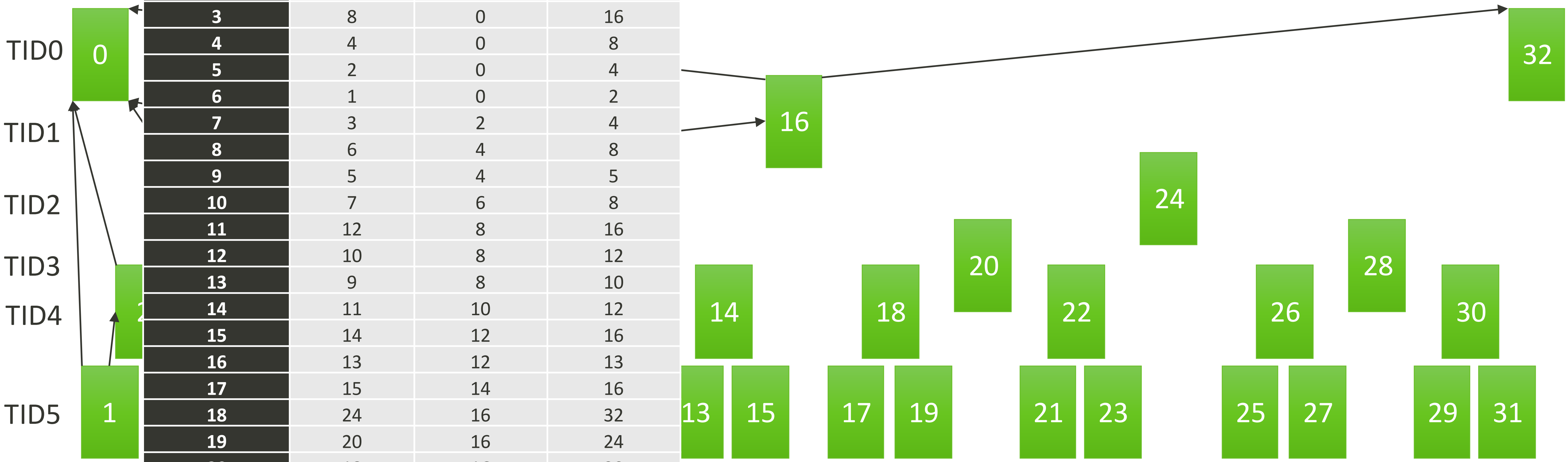
Hierarchical-B Coding Structure

- Can provide better coding efficiency than IPPP



- ility

Processing order	Target Picture	Reference Pic0	Reference Pic1
0	0 (X)		
1	32 (X)		
2	16	0	32
3	8	0	16
4	4	0	8
5	2	0	4
6	1	0	2
7	3	2	4
8	6	4	8
9	5	4	5
10	7	6	8
11	12	8	16
12	10	8	12
13	9	8	10
14	11	10	12
15	14	12	16
16	13	12	13
17	15	14	16
18	24	16	32
19	20	16	24
20	18	16	20



What You Need to Do

2. Please process the picture 1-31, 33-63, 65-95, 97-127 (no need to process picture 0, 32, 64, 96, and 128) and follow the Hierarchical-B processing order to process the video.
   - The available reference pictures are also defined
   - The target picture marked as “X” means that you don’t need to process this picture.
   - Please use the pre-defined reference pictures

Processing order	Target Picture	Reference Pic0	Reference Pic1
0	0 (X)		
1	32 (X)		
2	16	0	32
3	8	0	16
4	4	0	8
5	2	0	4
6	1	0	2
7	3	2	4
8	6	4	8
9	5	4	5
10	7	6	8
11	12	8	16
12	10	8	12
13	9	8	10
14	11	10	12

Processing order	Target Picture	Reference Pic0	Reference Pic1
32	31	30	32
33	64 (X)		
34	48	32	64
35	40	32	48
36	36	32	40
37	34	32	36
38	33	32	34
39	35	34	36
40	38	36	40
41	37	36	37
42	39	38	40
43	44	40	48
44	42	40	44
45	41	40	42
46	43	42	44

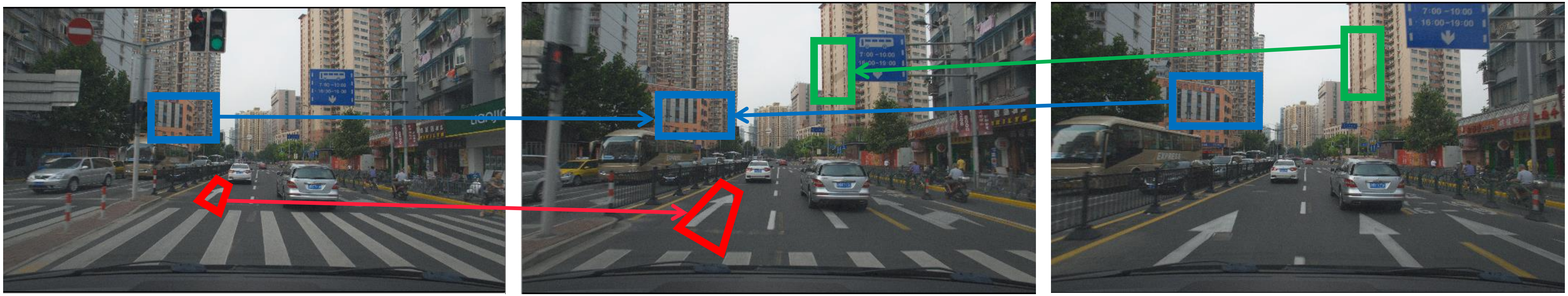
Processing order	Target Picture	Reference Pic0	Reference Pic1
96	95	94	96
97	128 (X)		
98	112	96	128
99	104	96	112
100	100	96	104
101	98	96	100
102	97	96	98
103	99	98	100
104	102	100	104
105	101	100	101
106	103	102	104
107	108	104	112
108	106	104	108
109	105	104	106
110	107	106	108

What You Need to Do

3. Divide each target picture into 16x16 blocks. **Please select 13,000 luma (Y) blocks for GMC.**
   - A picture contains 32,400 blocks. 13,000 blocks are roughly 40% of entire picture.
   - Only need to compensate the luma samples

What You Need to Do

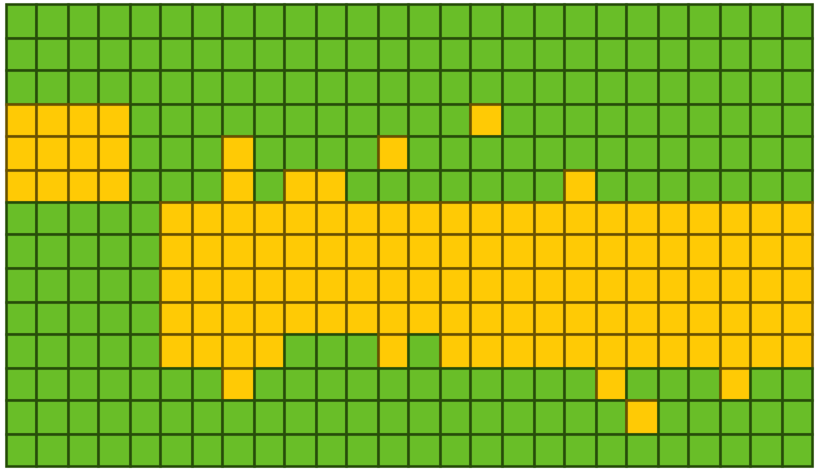
4. For a target picture, please derive at most 12 models that can compensate the 13,000 luma blocks of the target picture
   - One model can contain one affine/projection/perspective motion model from reference picture 0, one affine/projection/perspective motion model from reference picture 1, and one blending weight



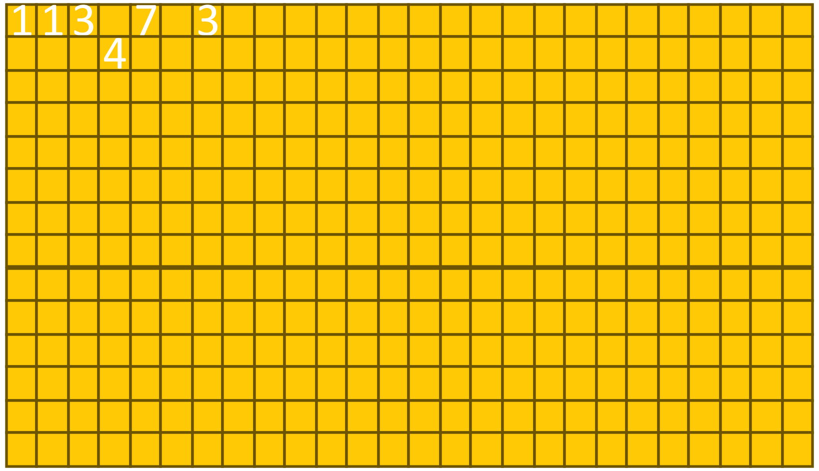
What You Need to Do

5. **Output the compensated monochrome video (luma only), selection map (which blocks are selected for compensation), model map (which model is used for the selected block), models (in document), and the complexity (MAC/pixel) of additional post-processing.**

Output image



Selection map
32400 bits



Model map,
32400x4 bits

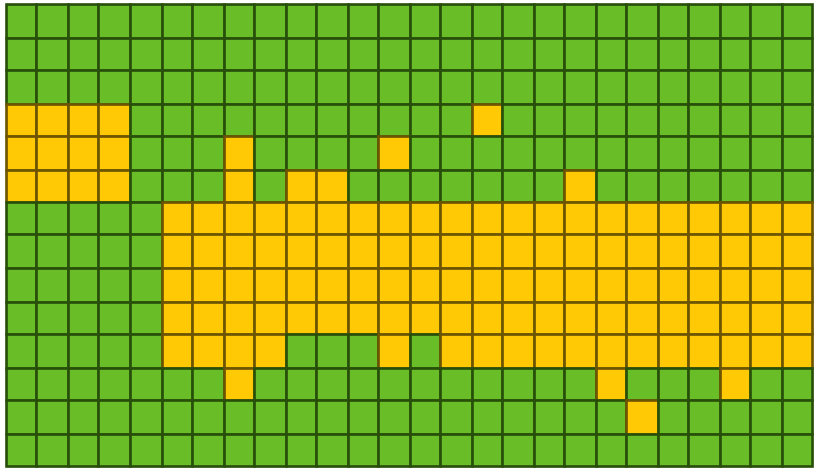
Hint

- 1. When doing the compensation, using interpolation filter to warp samples from reference picture may help. An example interpolation filter is shown in Table 2.
- 2. For each target picture, it has one reference picture before it and one reference picture after it. Blending two predictors from reference picture 0 and reference picture 1 may help.
- 3. You could perform segmentation to separate the foreground and background. Use different models for foreground and background, or even different objects.
- 4. Post processing could also help to improve the compensation quality. For example, de-blocking, over-lapped block motion compensation, illuminance compensation, pixel restoration network.
- 5. Use selection map to rule out the difficult blocks.

Hint – Selection Map

- Use selection map to rule out the difficult blocks
- Select best 13000 blocks for scoring

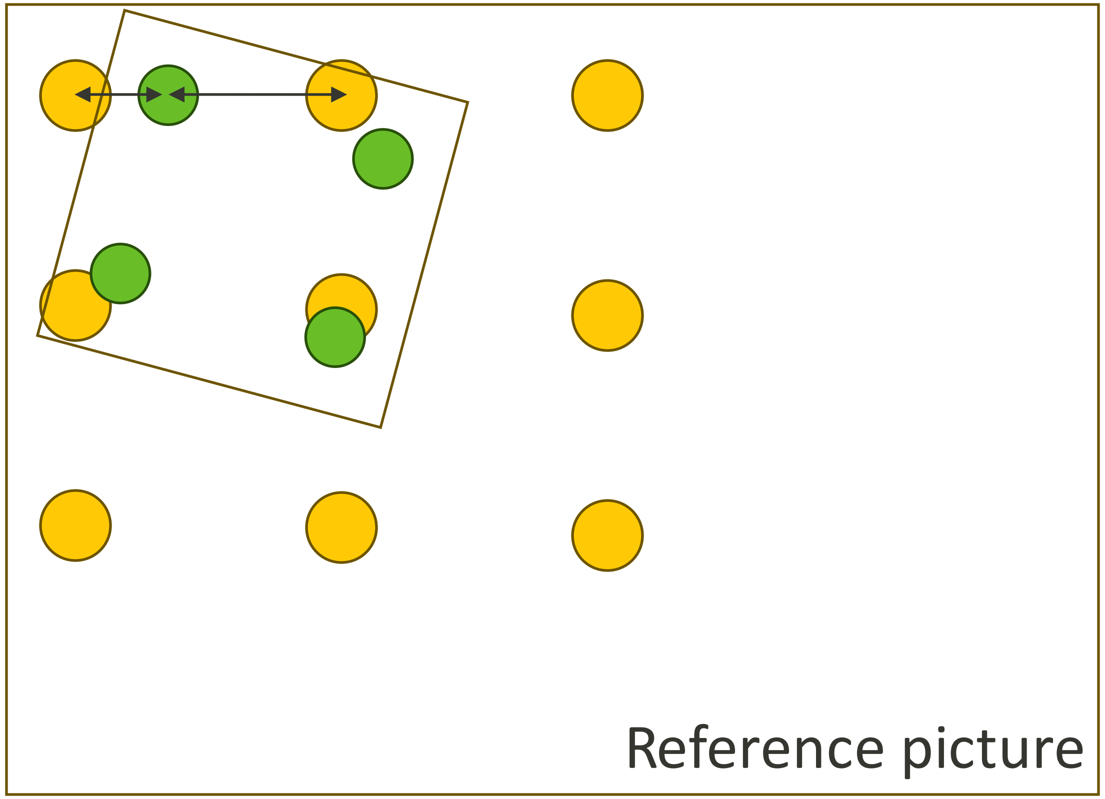
Output image

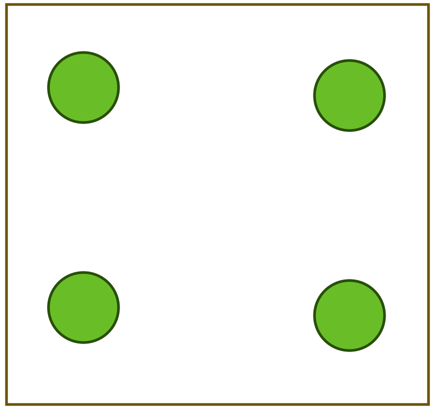


Selection map
32400 bits

Hint - Interpolation

- When doing the compensation, using interpolation filter to warp samples in fractional position from the reference picture may help.

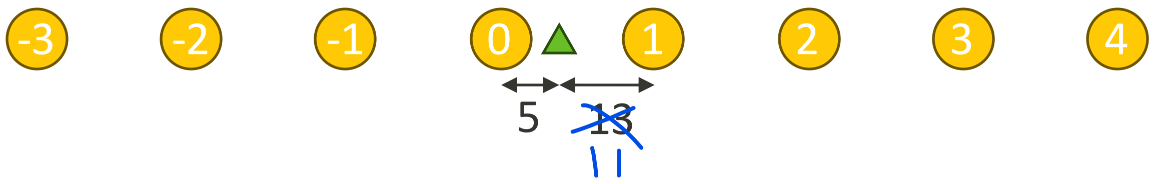




Current block

Hint - Interpolation

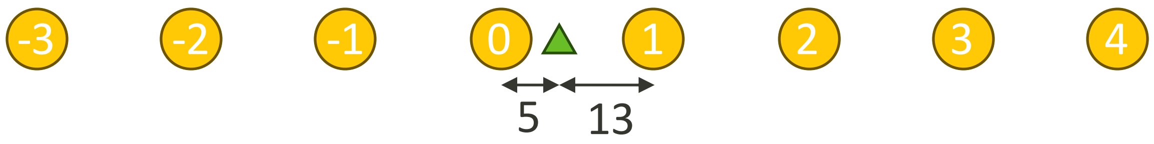
- **Here shows an 8-tap interpolation filter with 1/16-pel resolution**



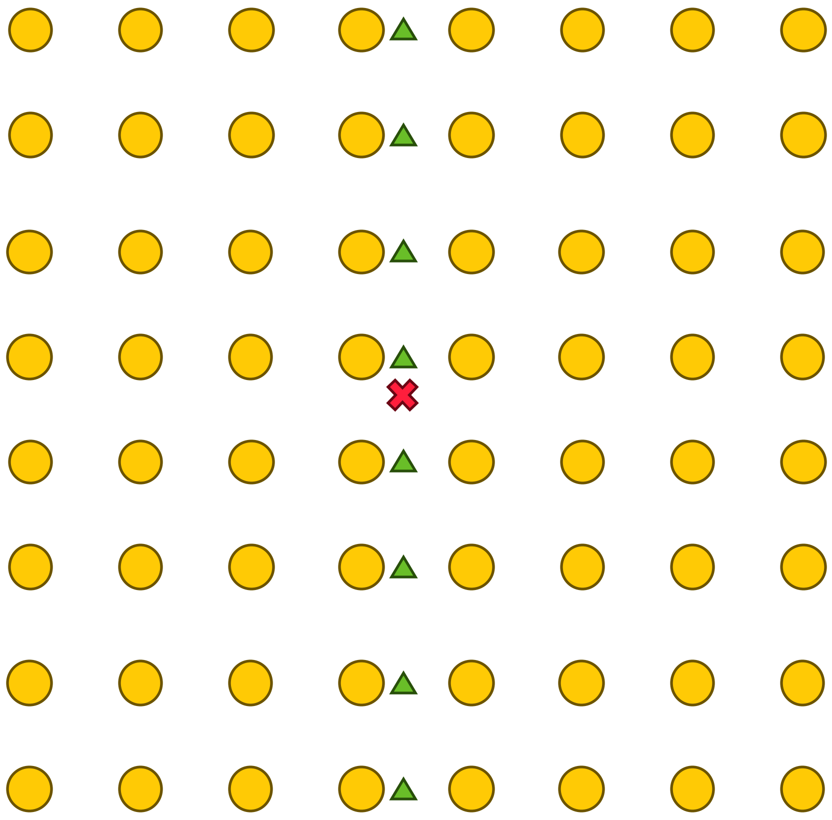
Fractional sample position p	interpolation filter coefficients							
	f[-3]	f[-2]	f[-1]	f[0]	f[1]	f[2]	f[3]	f[4]
0	0	0	0	64	0	0	0	0
1	0	1	−3	63	4	−2	1	0
2	−1	2	−5	62	8	−3	1	0
3	−1	3	−8	60	13	−4	1	0
4	−1	4	−10	58	17	−5	1	0
5	**−1**	**4**	**−11**	**52**	**26**	**−8**	**3**	**−1**
6	−1	3	−9	47	31	−10	4	−1
7	−1	4	−11	45	34	−10	4	−1
8	−1	4	−11	40	40	−11	4	−1
9	−1	4	−10	34	45	−11	4	−1
10	−1	4	−10	31	47	−9	3	−1
11	−1	3	−8	26	52	−11	4	−1
12	0	1	−5	17	58	−10	4	−1
13	0	1	−4	13	60	−8	3	−1
14	0	1	−3	8	62	−5	2	−1
15	0	1	−2	4	63	−3	1	0

Hint - Interpolation

- Here shows an 8-tap interpolation filter with 1/16-pel resolution

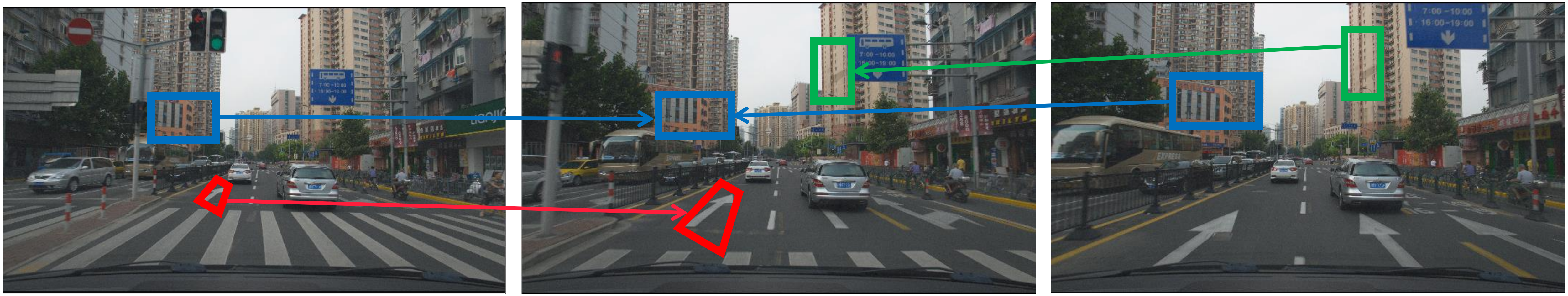


- Example of 2D interpolation



Hint – Blending from RefPic0 and RefPic1

- For each target picture, it has one reference picture before it and one reference picture after it.
- Blending two predictors from reference picture 0 and reference picture 1 may help.
  - The blue model uses bi-prediction

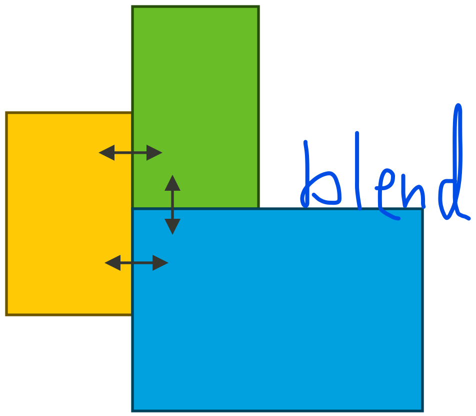


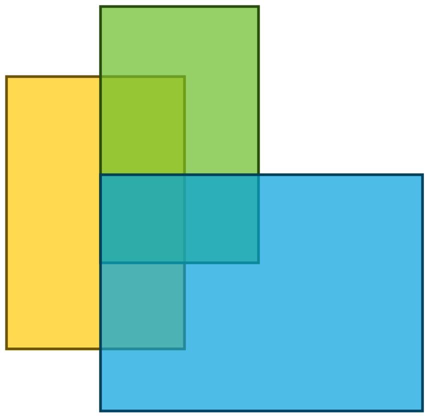
Hint – Segmentation

- You could perform segmentation to separate the foreground and background.
- Use different models for foreground and background, or even different objects.

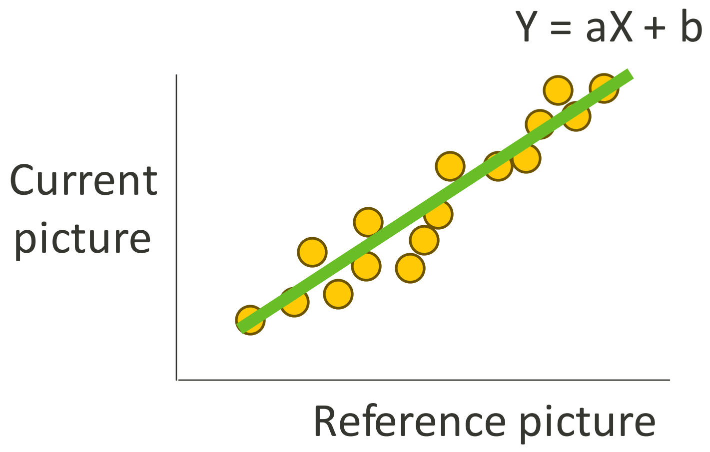
Hint – Post-processing

- Post processing could also help to improve the compensation quality. For example, de-blocking, over-lapped block motion compensation, illuminance compensation, pixel restoration network
- One model can contain one affine/projection/perspective motion model from reference picture 0, one affine/projection/perspective motion model from reference picture 1, one blending weight, **and one set of post-processing**.





OBMC



Illuminance compensation

Constraint

- Follow the Hierarchical-B processing order and the pre-defined reference pictures.
- At most 12 models can be used in a picture.
- One model can contain one affine/projection/perspective motion model from reference picture 0, one affine/projection/perspective motion model from reference picture 1, one blending weight, and one set of post-processing.
- Each selected block can only choose one model.
- Do NOT copy the golden data directly

Evaluation

- For each target picture, please select 13,000 blocks for compensation and calculating PSNR.
- According to submitted video and selection map, TA will calculate the average PSNR of the whole video sequence.
- $$PSNR = 10 \times log_{10}\left(\frac{255}{MSE}\right) \qquad MSE = \frac{\sum(Ori\ - Pred)^2}{Total\ Samples}$$

  , where MSE = mean-square error

Evaluation

- In this final project, it is allowed to apply post-processing to improve the compensation quality.
- However, in order not to use too complex post-processing, PSNR score deduction rule is list as below.
- Please report the complexity of your post-processing (if any) honestly.

Complexity	Final PSNR Score
< 100 MAC/pixel	No deduction. Final PSNR = PSNR
100 – 1K MAC/pixel	Final PSNR = PSNR – 0.05dB
1K – 20KMAC/pixel	Final PSNR = PSNR – 0.10dB
20K – 400KMAC/pixel	Final PSNR = PSNR – 0.15dB
> 400KMAC/pixel	Not allow

Evaluation - The criterion of score

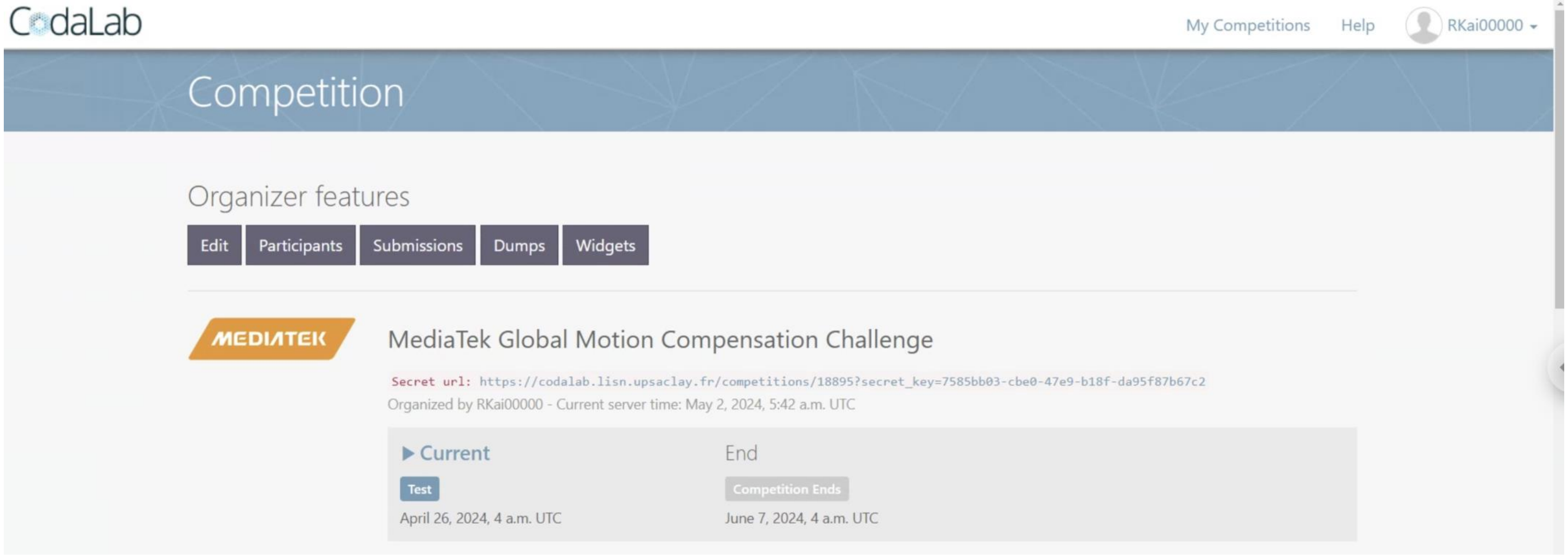
- **Objective quality: (50%)**
  - PSNR ranking
  - Constraints Violation
    - Not follow the Hierarchical-B processing order (-5%)
    - Use over 12 models in a picture (-5%)
    - Complexity > 400KMAC/pixels (-5%)
    - Copy golden data directly (-50%)
- **Presentation: (50%) Top 10 teams**
  - Novelty and technical contribution (20%)
  - Experiment completeness (25%)
  - Presentation (5%)
- **Report: (50%) Other teams**
  - Novelty and technical contribution (25%)
  - Experiment completeness (25%)

Points	# of teams
50%	1
48%	2
46%	2
42%	The rest teams /4
38%	The rest teams /4
34%	The rest teams /4
30%	The rest teams /4

MEDIATEK

Evaluation Server

- To be announced by TA



Project Output

1. The report of your final project
2. Compensated monochrome video (luma only)
3. Selection map (which blocks are selected for compensation)
4. Model map (which model is used for the selected block)
5. Description of models
6. Source code