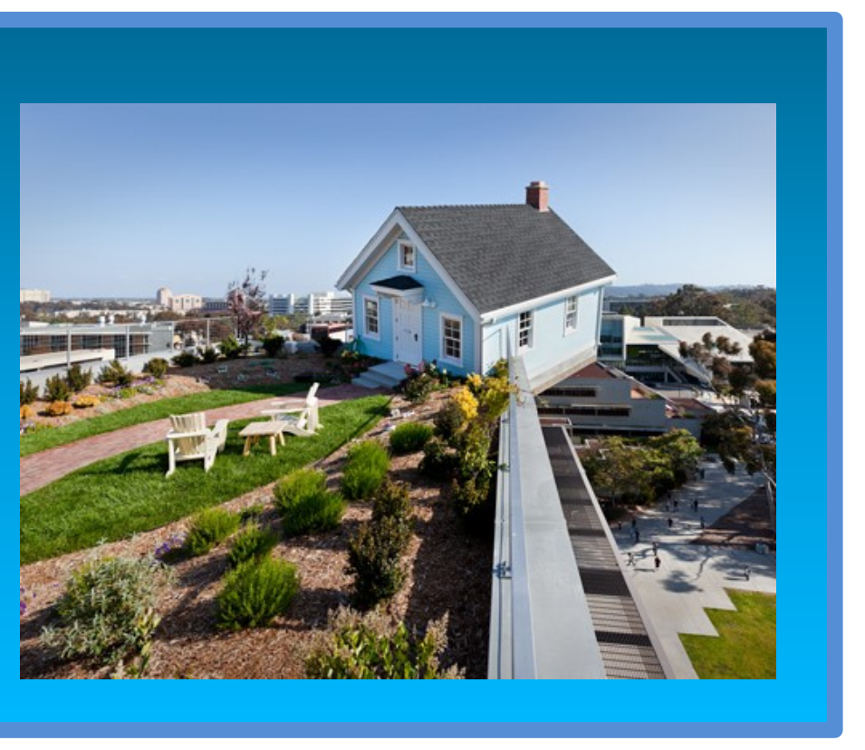
UC San Diego
Jacobs School of Engineering

PATCH-AWARE AVERAGING FILTER FOR SCALING IN POINT CLOUD COMPRESSION

Keming Cao[1], Yi Xu[2], Pamela Cosman[1]

[1]Dept. of Electrical and Computer Engineering, UC San Diego; [2] Owlii Inc., Beijing



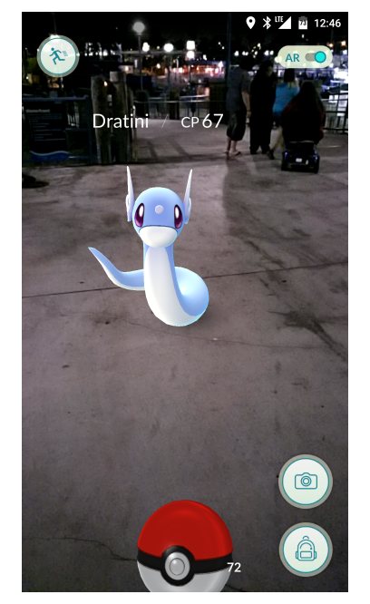
Abstract

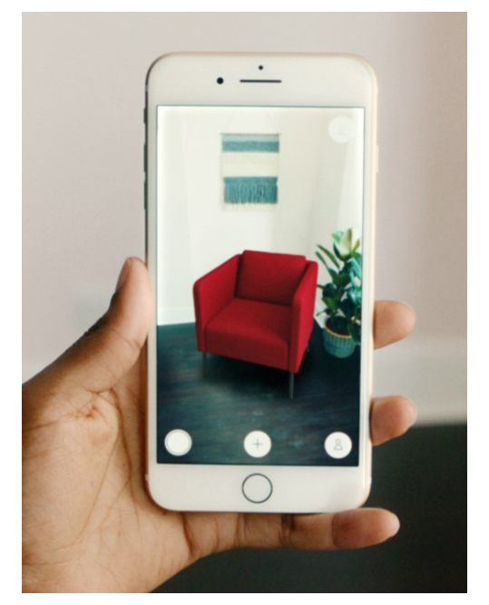
With the development of augmented reality, the delivery and storage of 3D content have become an important research area. Among the proposals for point cloud compression collected by MPEG, Apple's Test Model Category 2 (TMC2) achieves the highest quality for 3D sequences under a bitrate constraint. However, the TMC2 framework is not spatially scalable. In this paper, we add interpolation components which make TMC2 suitable for flexible resolution. We apply a patch-aware averaging filter to eliminate most outliers which result from the interpolation. Experimental results show that our method performs well both on objective evaluation and visual quality.

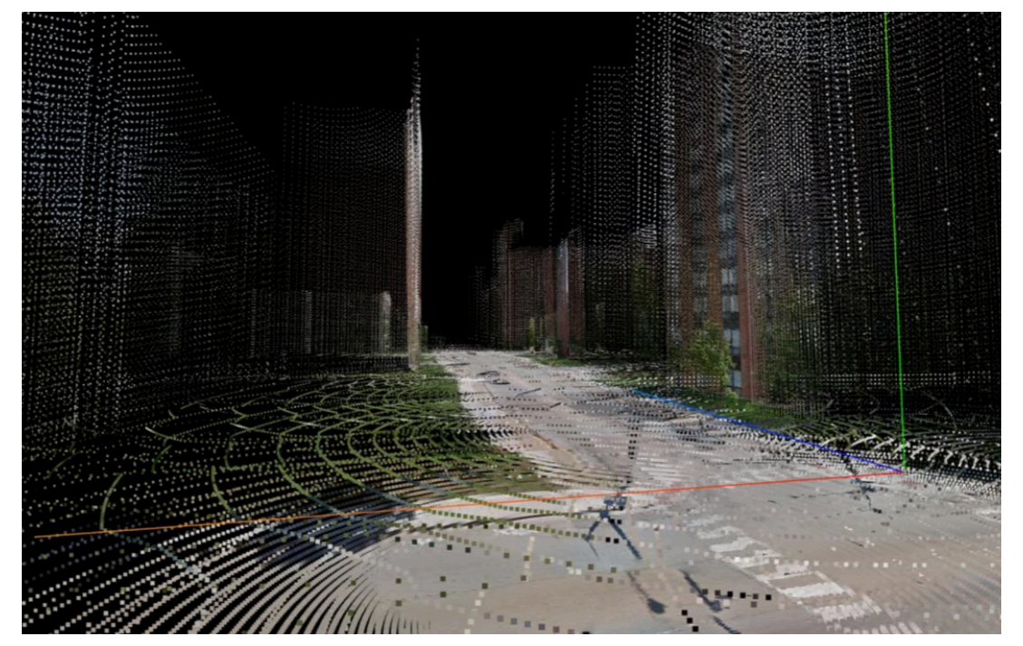
Motivation

3D media gets increasing attention

- Apps such as Pokémon Go, IKEA Place, Google Streetview, Amazon AR shopping
- Companies such as Google and Apple release platforms for smartphone-based AR/VR
- Investment reaches a new record high in 2017







Autonomous driving brings point cloud back

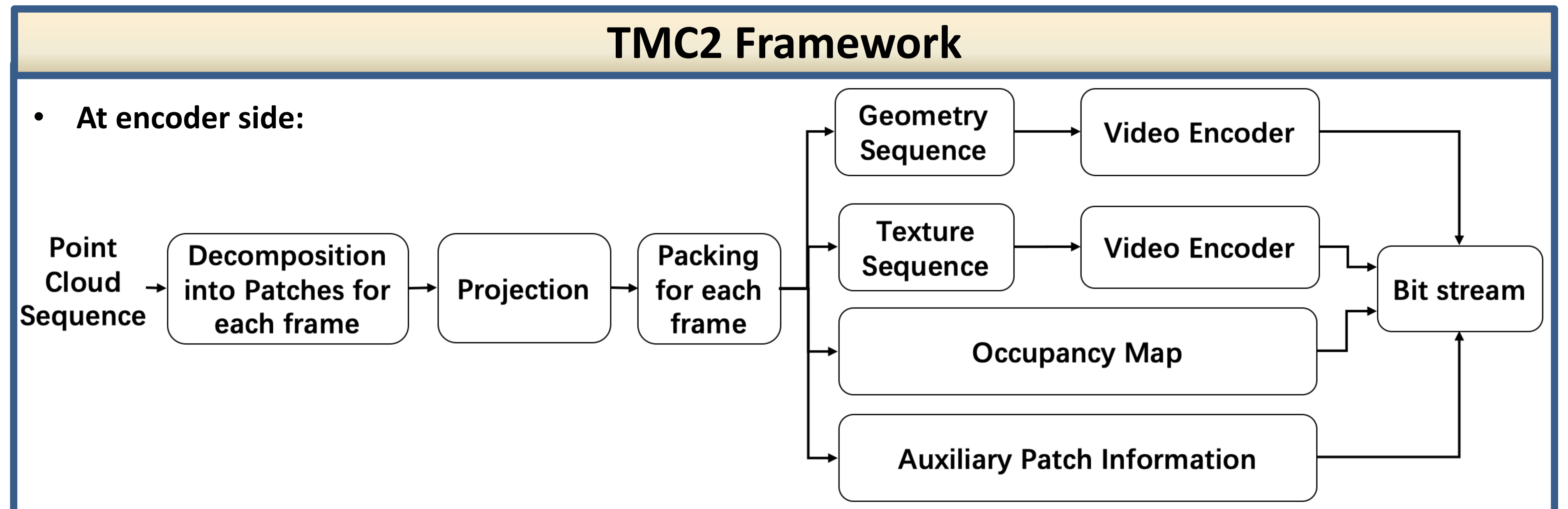
- Easy to capture from depth sensor
- Easy to concatenate

Compression standard is under research in MPEG

- Winner for point cloud sequence compression does not consider spatial scalability

TMC2 Framework

- **At encoder side:**

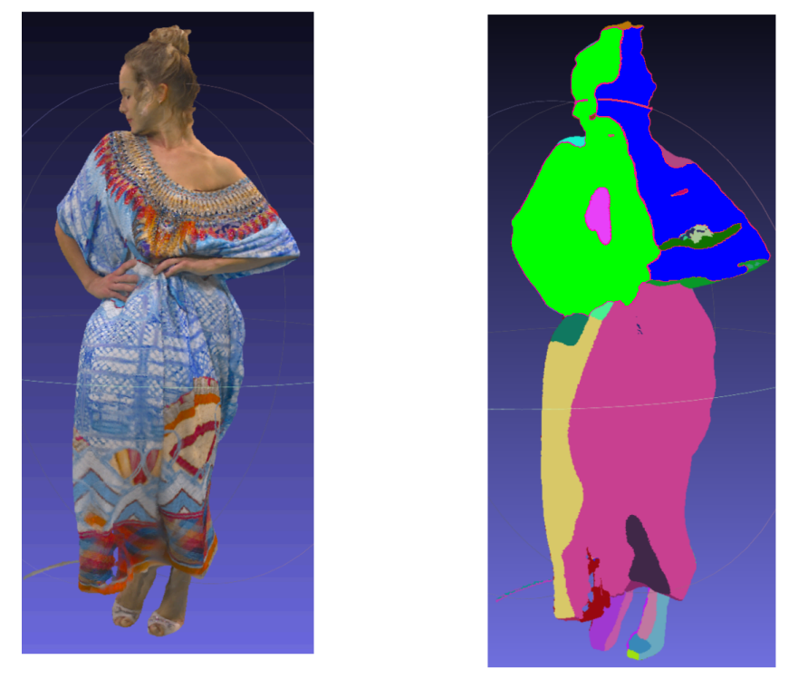


Decomposition (Segmentation):

① Computing normal for each point

② Clustering based on normal direction

③ Assigning each cluster a main direction (±x, ±y, ±z)

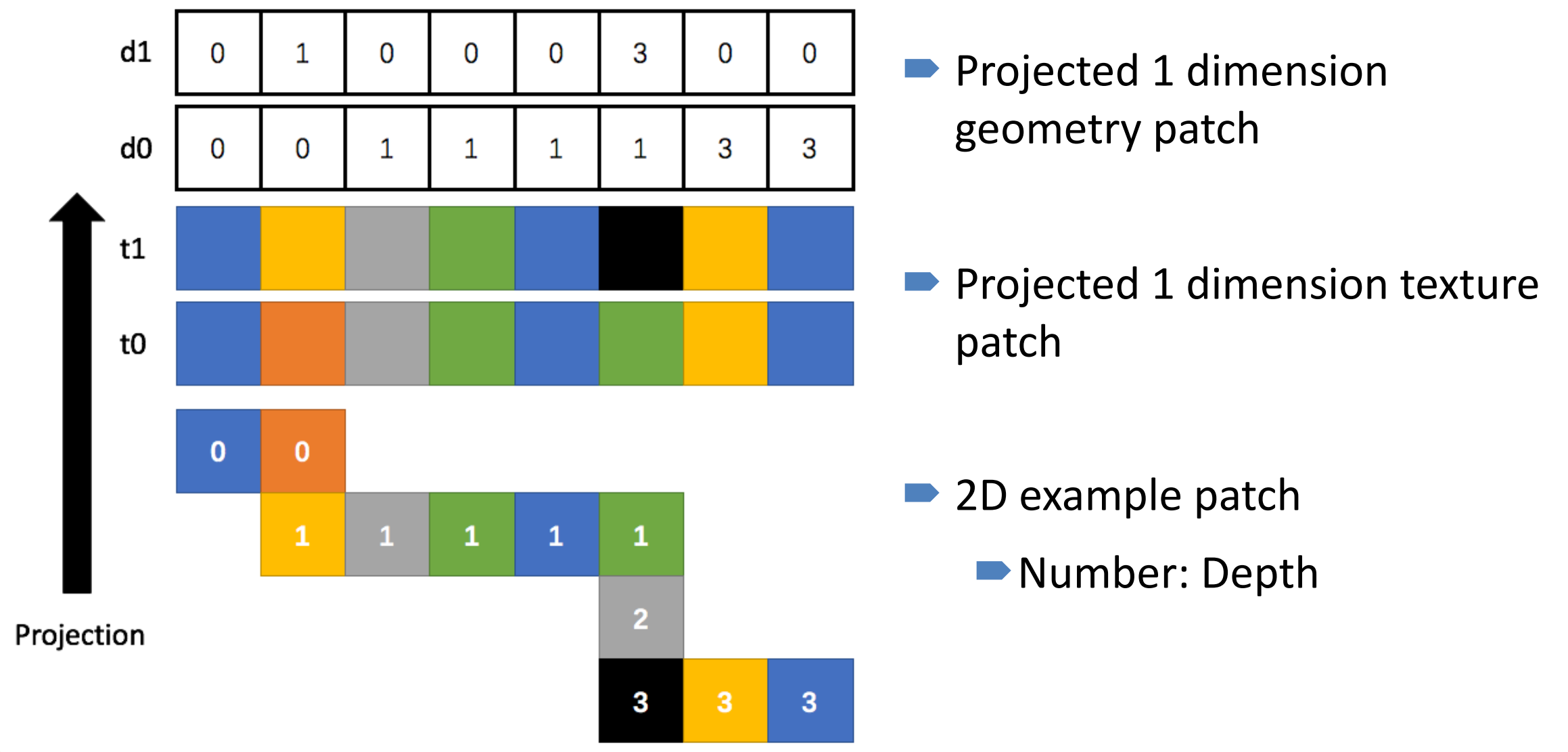


Different colors are different patches

Projection (Project from 3D to 2D):

- Project along main direction and project
- Project texture twice and geometry tiwce

2D patch example



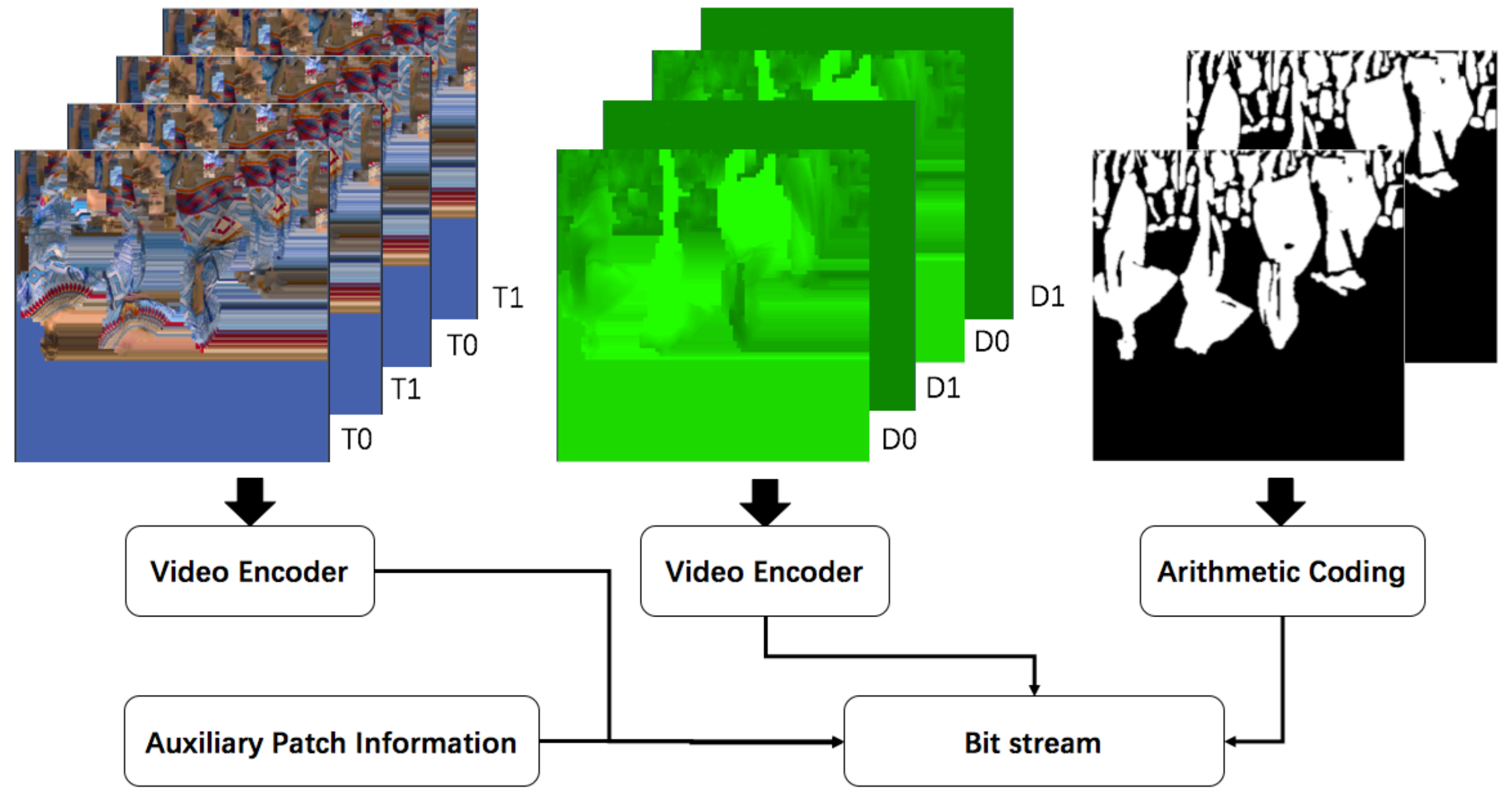
TMC2 Framework

Packing:

- Fitting all projected patches onto an image of pre-fixed size $l_x \times l_y$
- Occupancy map to record whether a pixel is occupied or not
- Pixels between patches are filled with value of nearby pixels
- Packing image is shown below (for texture, for geometry and occupancy map)

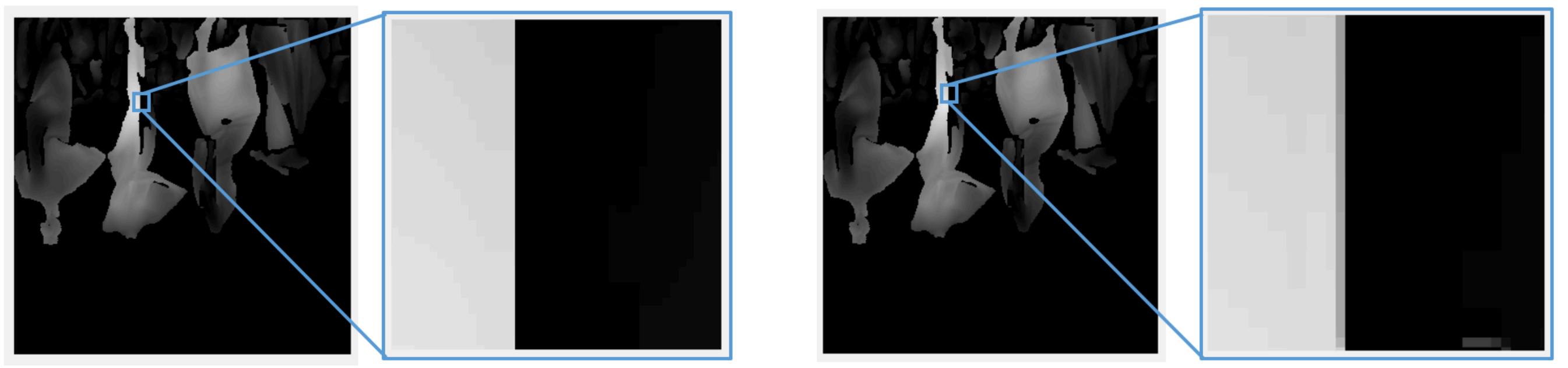
Encoding:

- Treat projected image sequence as 2D video sequence (both texture and depth)
- Video encoder could be any 2D video codec
- Occupancy map is encoded with arithmetic coding



Patch-aware Averaging Filter

- Add scale-down module at encoder side and scale-up module at decoder side
- Simple scaling module will cause noise at edge of patch

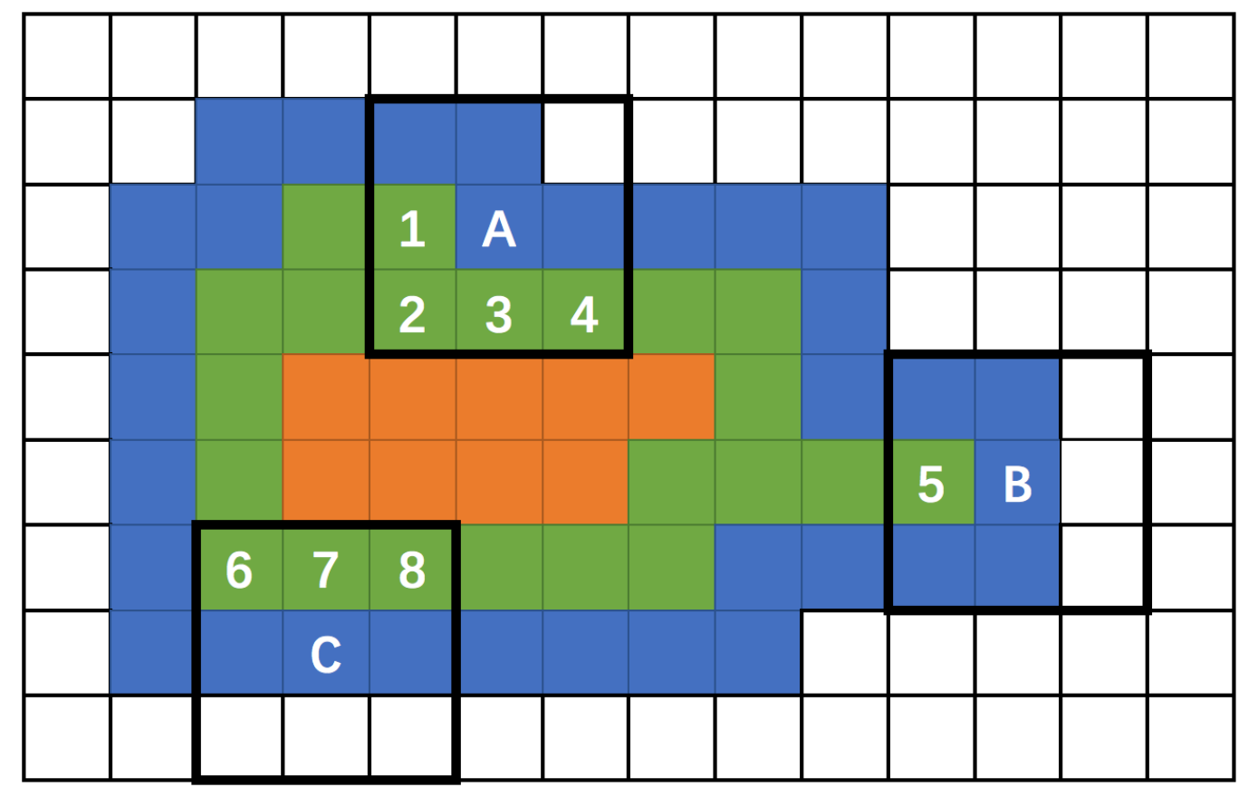


Original geometry image

Geometry image after scale-down and scale-up

- Patch-aware averaging filter is added after scale-up module at the decoder side
- Pixels in 2D geometry patch p_i are divided into 3 sets:

S_{n1}^i: pixels of chessboard distance 1 to pixels outside the patch
S_{n2}^i: pixels of chessboard distance 2 to pixels outside the patch
S_b^i: remaining pixels of the patch



- Pixel values in set S_{n1}^i are modified by:

$$I'_{up}(x,y) = \begin{cases} \frac{1}{M}\sum_{m\in Q} I_{up}(x_m, y_m), & Q \neq \emptyset \\ I_{up}(x,y), & Q = \emptyset \end{cases}$$

Q is the set of points that belong to S_{n2}^i and have chessboard distance 1 to (x,y), M is the size of Q, $I_{up}(x,y)$ is pixel value for (x,y)

Evaluation

- Take 12 frames from four sequences, *basketball*, *dancer*, *model*, *exercise* (2.5 million points per frame)
- Evaluate under: scale factors (S), quantization parameters (QP), interpolation combinations (C) and different filter types (F)

Evaluation

Evaluation metric

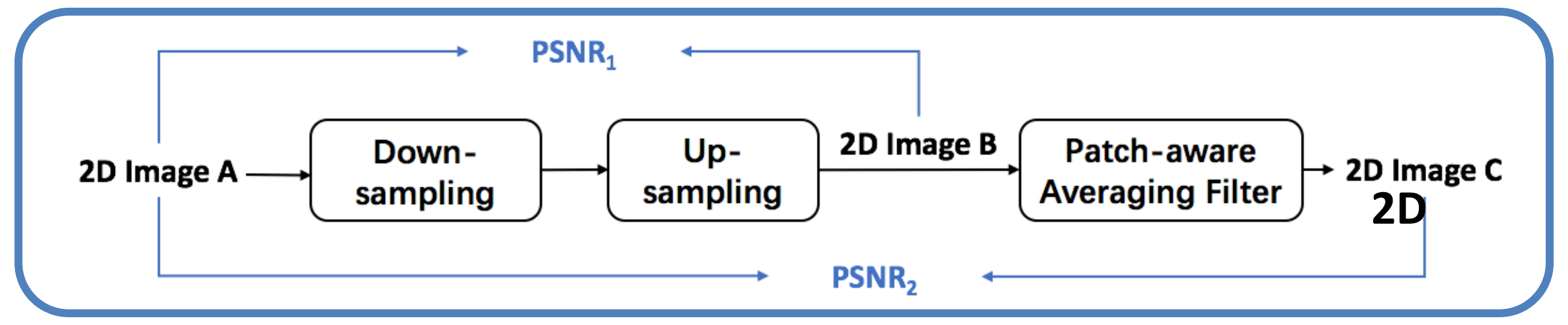
- 2D: PSNR
- 3D: MSE defined below

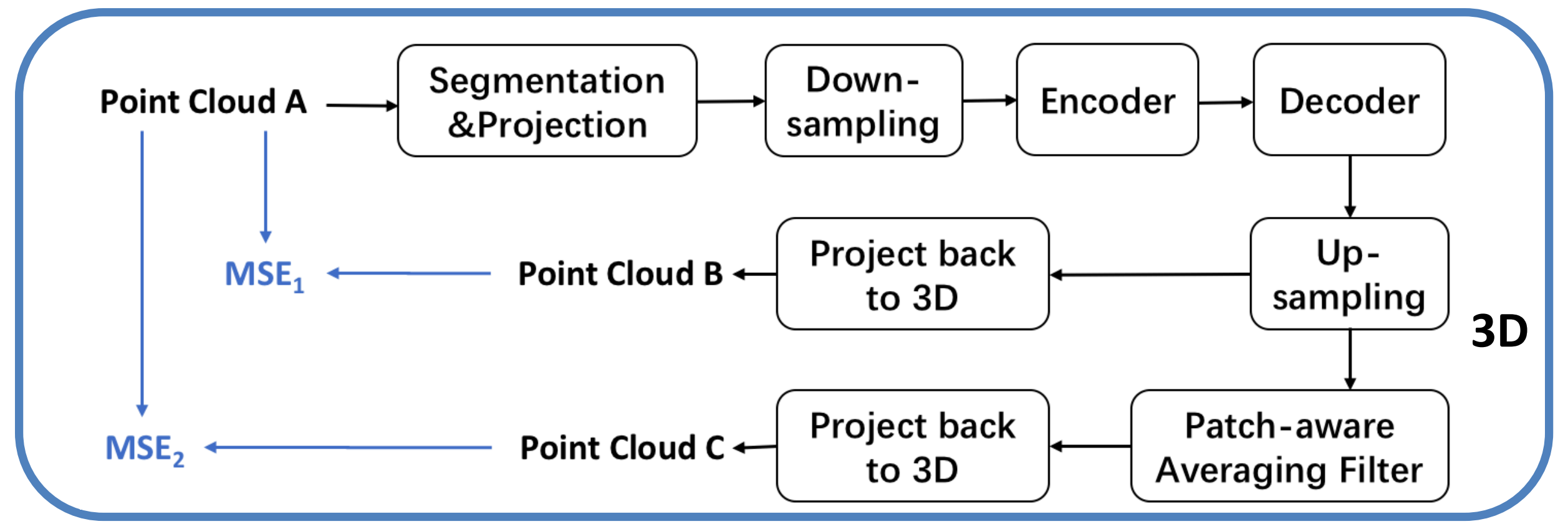
$MSE_1 = \max(MSE_{AB}, MSE_{BA})$

where $MSE_{AB} = \frac{1}{N_A}\sum_{p_i \in A} |e_{AB}(i,j)|^2$,

$$MSE_{BA} = \frac{1}{N_B}\sum_{p_i \in B} |e_{BA}(j,i)|^2$$

$e_{AB}(i,j)$ is error vector from point p_i in A to the closest point p_j in B, N_A is number of points in A.





Interpolation combination:

- NN2NN: Nearest neighbor at the encoder and decoder
- NN2B: Nearest neighbor at the encoder and bicubic at the decoder
- B2NN: Bicubic at the encoder and nearest neighbor at the decoder
- B2B: Bicubic at encoder and decoder

Scale factors:

- 1.25, 1.5, 1.75, 2.0, 2.25, 2.5

Quantization parameters:

- 0, 10, 20, 30, 40

Filter types:

- Averaging filter (A), Median filter (M), Weighted filter (W) with 3x3 Gaussian kernel

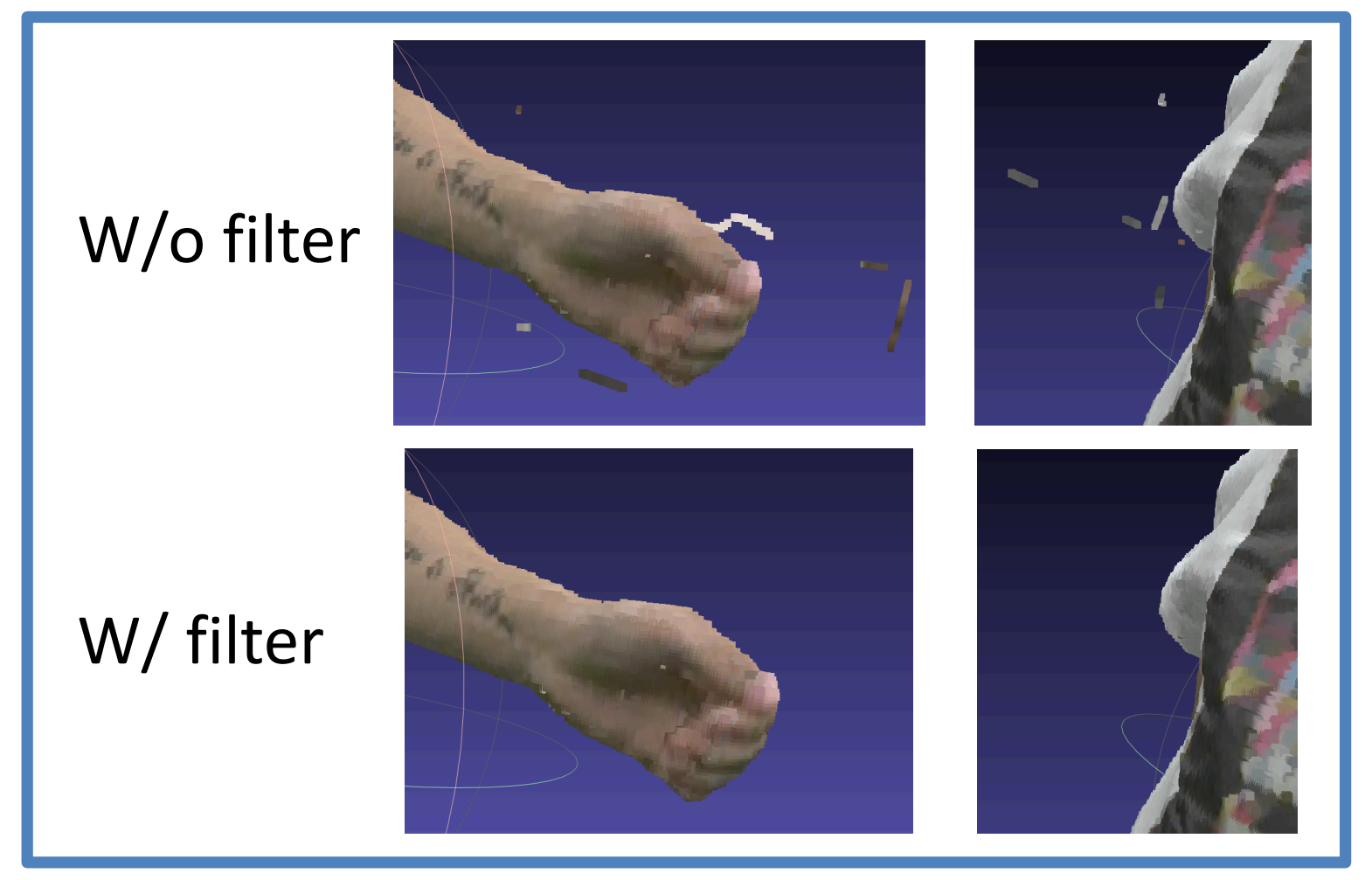
- **Evaluate for different S and C**

$$\Delta_{PSNR} = PSNR_2 - PSNR_1$$

Table 1. Δ_{PSNR}

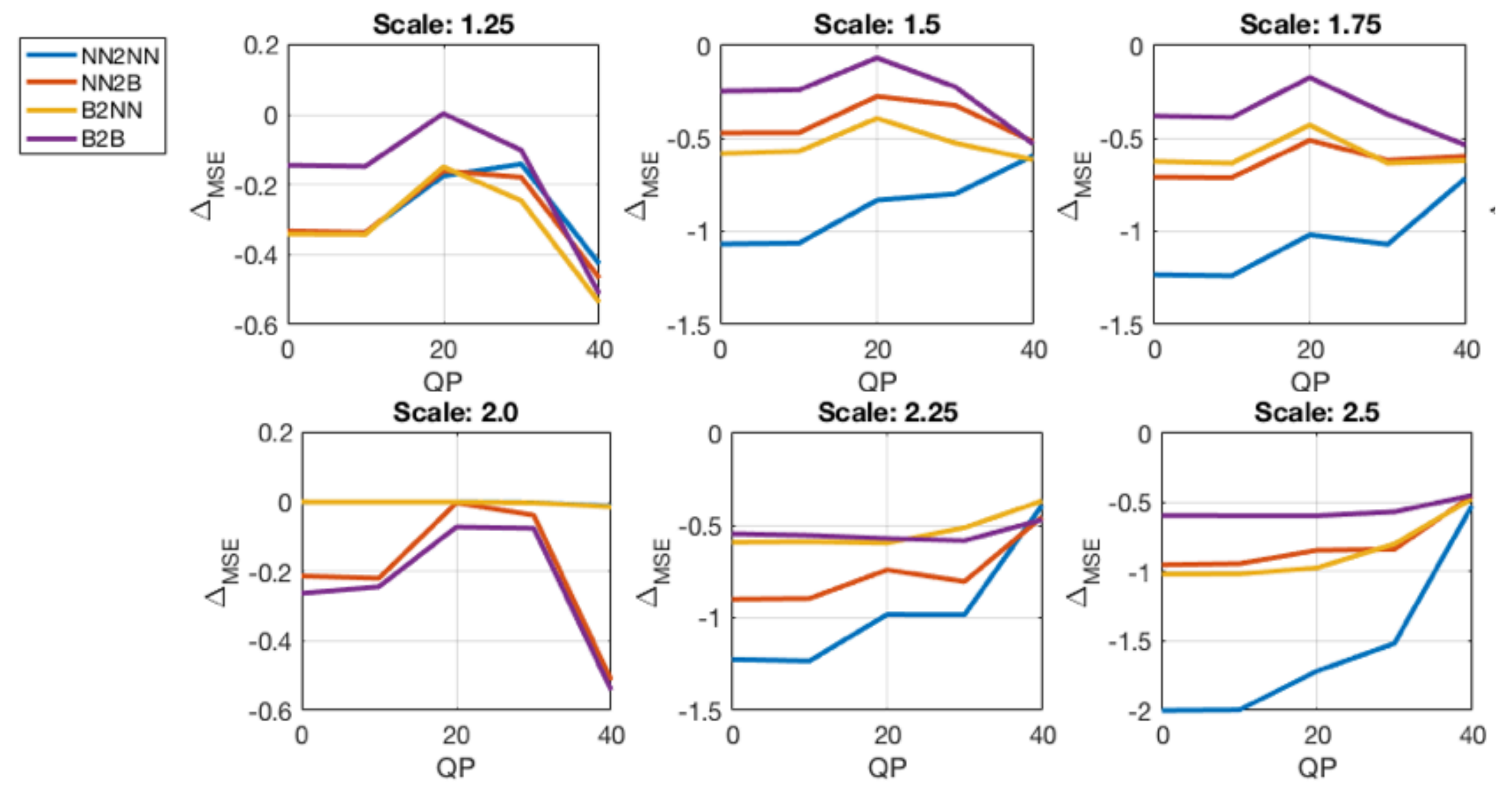
Factors	1.25	1.5	1.75	2.0	2.25	2.5
Setup NN2NN:						
dancer	5.46	5.24	7.23	-0.01	7.33	7.18
basketball	4.66	6.8	7.80	0.01	7.42	8.17
model	5.30	5.6	7.02	-0.13	6.46	7.61
exercise	6.42	6.39	7.84	-0.20	8.12	8.15
Setup NN2B:						
dancer	7.99	8.2	7.59	4.19	5.47	4.58
basketball	8.42	9.01	8.28	4.93	5.82	4.66
model	8.23	8.24	7.29	4.50	5.11	4.66
exercise	9.68	9.26	8.29	5.30	5.72	5
Setup B2NN:						
dancer	6.82	6.16	6.69	0.04	5.25	4.69
basketball	7.48	7.53	7.63	0.07	5.14	4.73
model	6.49	6.40	6.89	-0.09	4.39	4.93
exercise	7.34	7.19	8.02	-0.07	5.6	5.31
Setup B2B:						
dancer	6.92	7.65	7.51	7.37	5.72	4.70
basketball	7.57	8.40	8.08	8.36	6.14	4.91
model	6.69	7.45	7.16	7.30	5.40	4.76
exercise	7.58	8.46	8.19	8.49	5.80	5.25

$\Delta_{PSNR} > 0$ denotes higher PSNR with filter



- **Example for sequence *dancer* with different S, QP and C**

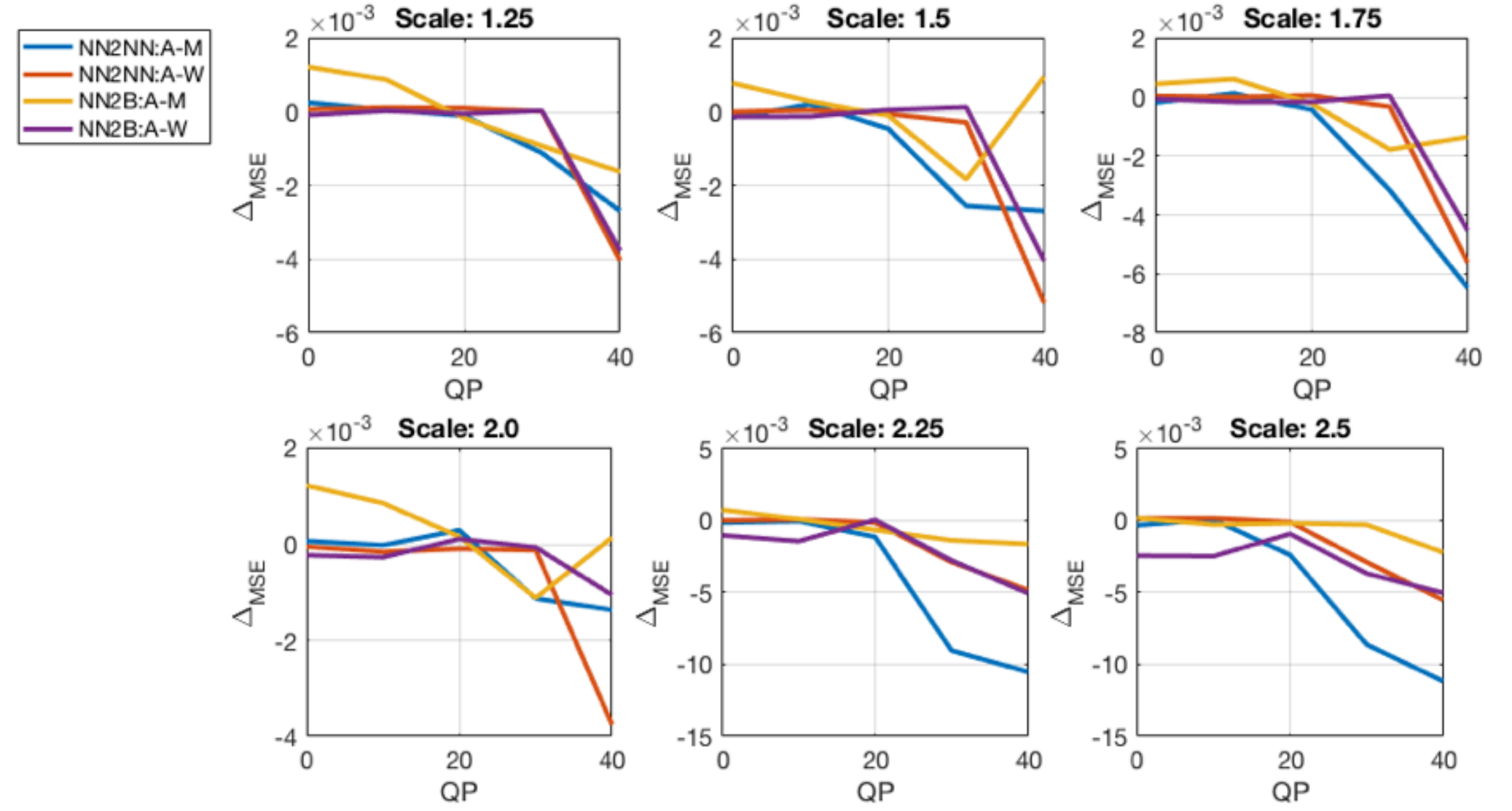
$$\Delta_{MSE} = MSE_2 - MSE_1$$



$\Delta_{MSE} < 0$ denotes smaller MSE with filter

- **Compare different F**

A-M: $\Delta_{MSE} = MSE_A - MSE_M$, A–W: $\Delta_{MSE} = MSE_A - MSE_W$



The averaging filter generates slightly lower MSE than the median filter and a weighted filter. The experimental results on 2D PSNR and 3D MSE as well as visual inspection of image pairs show that our method performs well both on objective evaluation and on subjective visual quality.