# Inter-prediction for Multiview Video Coding
# Final Project Report

## I. Project Introduction

### 1.1 Backgrounds

With the increasing demand for 3D video and the advance of multimedia and camera related technologies, Multiview video coding attracts more and more attention and plays a more significant role within these years.[1] Multiview video can give views a whole new experience by providing combined information of the scene structure from different viewpoints.[2]

Practical applications[3] such as virtual reality (VR) and 3D object construction and motion estimation have promising prospects and can further boost the development of light field imaging theories. However, there still exists many other problems that hinder further application for Multiview video coding. For example, Multiview video is a big volume of data, which is required to be compressed.

### 1.2 Problem description

In this project, we are given 7 groups of images RGB form with intensity varies in [0, 255]. These images are captured by 9 cameras that are distributed in a 3×3 arrangement as shown in Fig.2. Each group contains 18 images taken in two frames, 9 for each frame. The images are named in order of camera sweep rules as shown in Fig.3.

Fig.1 Names of given datasets

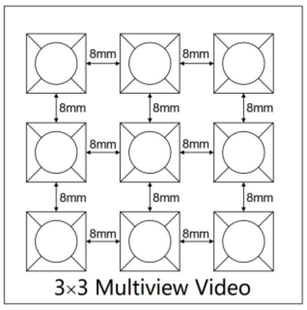


Fig.2 Arrangement of cameras

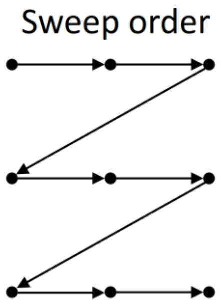


Fig.3 Camera sweep order

Pictures are named ‘R1’, ‘R2’, …, ‘R9’ in frame 1 and ‘R10’, …, ‘X’, …, ‘R17’ in frame 2, and ‘R5’ and ‘X’ are center images in frame 1 and frame 2 respectively.

Our task is to propose a convex optimization model and predict the image ‘X’ based other 17 reference images. The predicted image should be as similar as possible with the original image X.

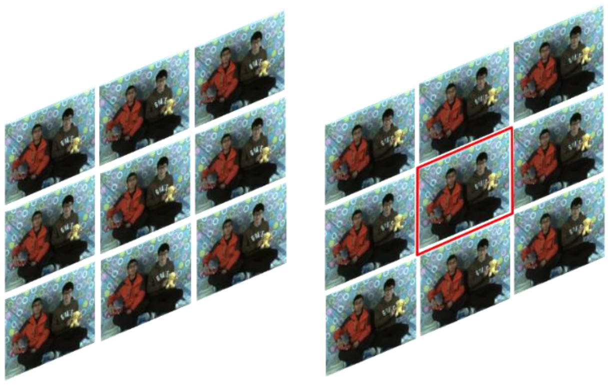


Fig.4 Images of frame 1 (left) and images of frame 2 (right)

(Image in the red frame is the one we need to predict)

## 1.3 Project requirements

There are certain requirements we need to be aware of:

1. The original to-be-predicted image X cannot be used in the optimization model;

2. More than one temporal reference image (R1 – R9) should be applied into the optimization model;

3. More than one spatial reference image (R10 – R17) should be applied into the optimization model;

4. The predicted image should be segmented into small predicted blocks;

5. The size of each block must be 4×4, 8×8, 16×16, 32×32 or 64×64 as shown in Fig.5;

6. The reference blocks must have the same size with predicted block;

7. The reference blocks may have different location with the predicted block as shown in Fig.6.

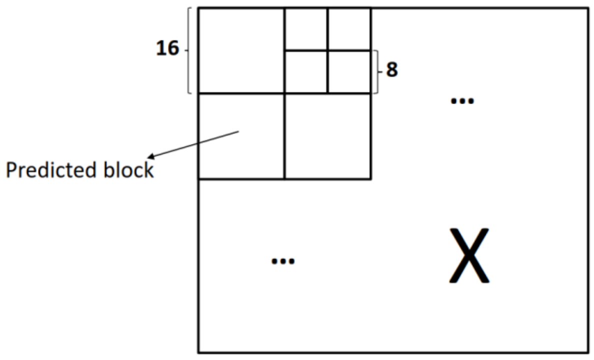


Fig.5 Illustration of block size

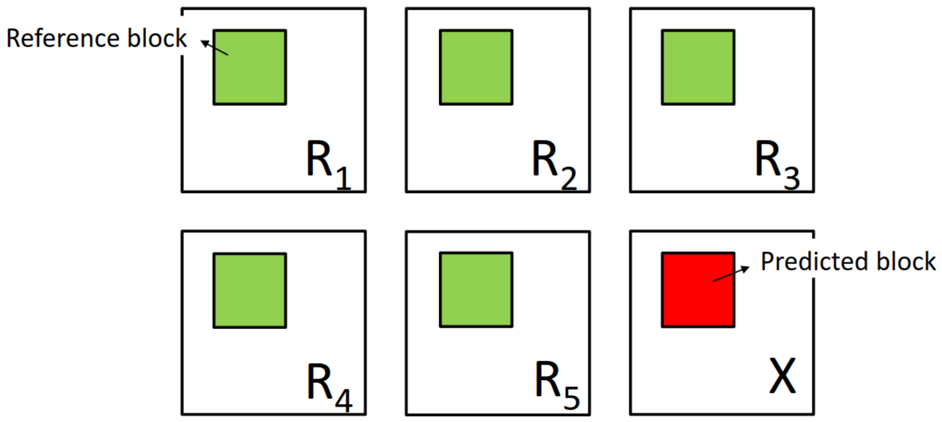


Fig.6 Illustration of block location

8. The evaluation rule is the mean PSNR of three channels of the predicted image as below:

$$PSNR_{mean} = \frac{1}{3} \times \left( PSNR(X_R, X'_R) + PSNR(X_G, X'_G) + PSNR(X_B, X'_B) \right) \quad (1)$$

9. Our convex optimization model should generate a predicted image in RGB format with intensity varying in [0, 255];

10. We should calculate the PSNR of the original and predicted image of each dataset and the average PSNR of seven datasets.

11. We should generate a partition image to show the partition result of the predicted blocks comprising predicted image;

12. We should upload our source code of the algorithm with cost evaluation;

13. We should write a project report including necessary contents that are related to the project.

14. Presentation slides are required.

### 1.3 Work Division

王萧诚：Basic mode, object segmentation, improved model, report & slides, code optimization.

欧阳湘凯：Deviation correction, improved model, slides & report, further improvement (camera offset correction).

## II. Problem Analysis and Modeling

### 2.1 Problem analysis

Directly speaking, this is an image prediction problem and should be an easy one since we are given 17 reference images to predict one image. However, the difficulties we face are more than that. Generally, there are several things we should consider:

1. The geometry relationship of cameras and objects captured by cameras;
2. The possible translation, shear, rotation transformation of certain objects between two frames;
3. The possible change in cameras' position and relative location;
4. There could be depth information in certain images.

Taking above tips into account, we first analyze these 7 groups of images and see how they change between two frames by visualize the difference between images captured by the same camera. There are 8 difference images in each group since we are not allowed to use the center image of frame 2.

Table.1 Difference analysis

| Dataset | Difference analysis |
|---|---|
| NagoyaOrigami | Almost no change |
| NagoyaDataLeading | Almost no change |
| NagoyaFujita | The green mat spinning clockwise a little |
| Cam_Still | The right knight moving right a little |
| Toys | The red car moving right a little |
| Trees | The whole scene moving right a little |
| Boys | Both boys moving arms a lot |

From above we can find that some of the given datasets have very little change but others show various changes. Here we give several examples for further analysis and all detailed figures can be found in attachment folder /Pictures.

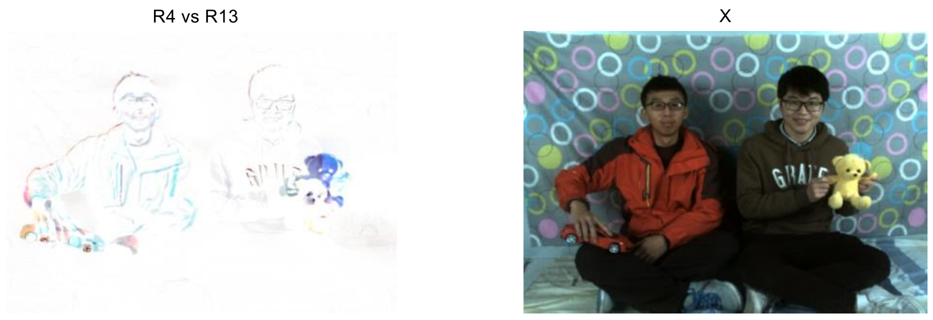


Fig.7 Selected difference of Data/Boys

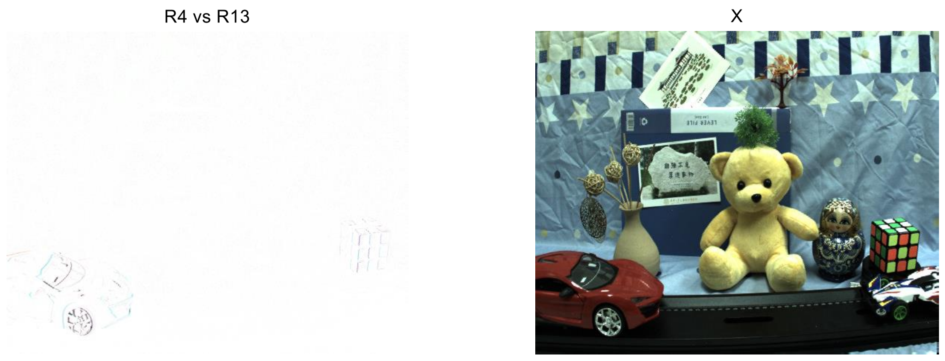


Fig.8 Selected difference of Data/Toys

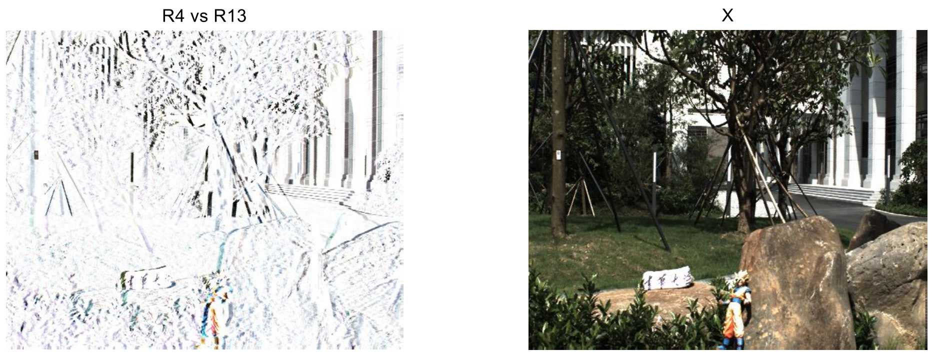


Fig.9 Selected difference of Data/Trees

From above three pictures we can clearly find that datasets like Boys and Toys show significant difference only in moving objects (boys' arms and toy car) while datasets like Trees show significant difference in the whole scene, or more specifically, the background.

From the difference analysis we can discover a hint that in some datasets, the background does not change between two frames, that is to say, the location of cameras does not change. On contrary, in other datasets, location of cameras changed between two frames thus lead to the difference in backgrounds shown in images.

Differences still exist between images captured by different cameras in one frame. Since 9 cameras are located in chess-board distribution, the same object will certainly exhibit in different location in 9 images captured by corresponding cameras.

To conclude, in later modeling section, we should consider 2 kinds of differences:

1. Spatial differences between 9 images in the same frame due to the location of cameras and light condition;

2. Temporal differences between 2 images captured by same camera in two frames due to changes of objects, location of cameras and light condition.

## 2.2 Convex optimization models

### 2.2.1 Basic model (Spatial Difference Model)

Before modeling, we need to implement some pre-processing to given datasets since we need to satisfy specific requirements mentioned in section 1.3.

Since the block size can be chosen from 4×4, 8×8, …, 64×64, we need to add rows and columns of given images so that they can be exactly divided by block size with zero remainder. The pseudo code of this procedure is shown below:

```
[row, col] = size(Img)
if mod(row, block_size) != 0
    row_add = block_size – mod(row, blocksize)
else
    row_add = 0;
if mod(col, block_size) != 0
    col_add = block_size – mod(col, blocksize)
else
col_add = 0;
Img(row+1 : row+row_add, :, :) = 0
Img(:, col+1 : col+col_add, :) = 0
```

We separate images into 3 channels (*R, G, B*) and name these images as *img_xx_C*, where *xx* can be *ul* (up-left), *uc* (up-center), *ur* (up-right), *ml* (middle-left), *mc* (middle-center), *mr* (middle-right), *dl* (down-left), *dc* (down-center), *dr* (down-right) and *C* can be *R*, *G*, *B*.

We propose a basic model by just focusing on the spatial differences *img1_xx_diff_C* (same name law as above) under the assumption that objects in surrounding images do not change too much between two frames. This is a very primitive idea by observing the given datasets. The scene is shown in Fig.10.

Specifically, we compute 8 difference matrices by subtracting 8 surrounding images from the center image in frame 1, then add the difference matrices to corresponding images in frame 2, finally optimize the weighted sum of *Frobenius-norm* distance between predicted image with the images after adding operation as shown in Fig.11.

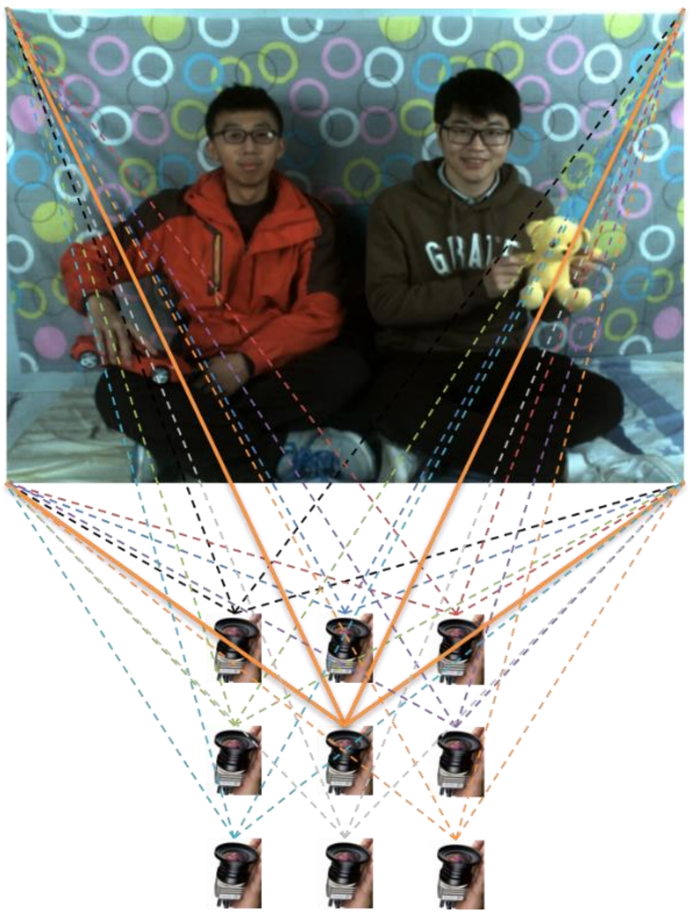


Fig.10 Working principle of multi-camera

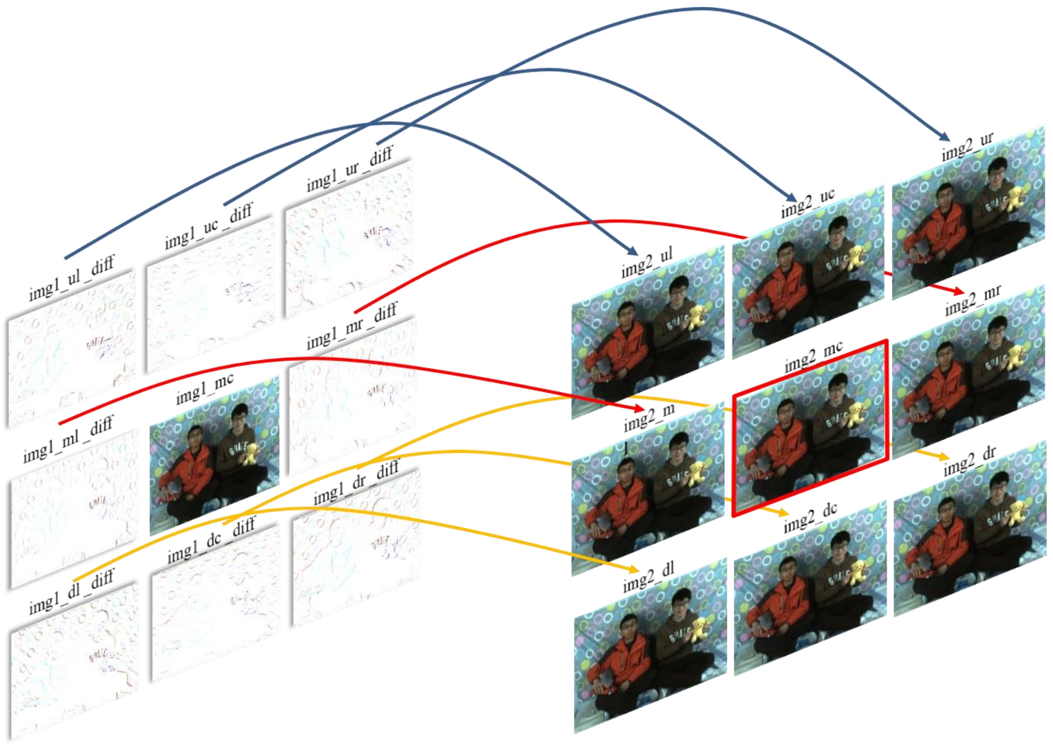


Fig. 11 The principle of basic model

From above illustration we can deduce the formal description of our basic model:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{n=1}^{8} \alpha_n \| X_{pred} - Ref_n \|_F \\
\text{subject to} \quad & X_{pred}(i,j) \geq 0; \\
& X_{pred}(i,j) \leq 255; \\
& Ref_n = img2\_xx + img1\_xx\_diff; \\
& img1\_xx\_diff = img1\_mc - img1\_xx; \\
& \alpha_n \geq 0; \alpha_n \leq 1; \\
& \sum_{n=1}^{8} \alpha_n = 1; \\
& i = 1,2,\ldots,block\_size; \\
& j = 1,2,\ldots,block\_size \\
& n = 1,2,\ldots,8; \\
& xx = ul, uc, ur, ml, mr, dl, dc, dr.
\end{aligned} \tag{2}
$$

Remember that we do not optimize the whole image at once but divide the image into many blocks and then implement optimization for *R, G* and *B* channel. The reason why we us *Frobenius-norm* is that we find it show better performance than other norms in last experiment (upsampling task).

### 2.2.2 Deviation correction

The basic model takes spatial differences into account but still not good enough since it doesn't consider the translation between cameras and the resulting translation of object positions in images.

Hence, we can improve the basic model by computing the deviation of surrounding images and correcting them. After we get the corrected images, we can then implement the optimization.

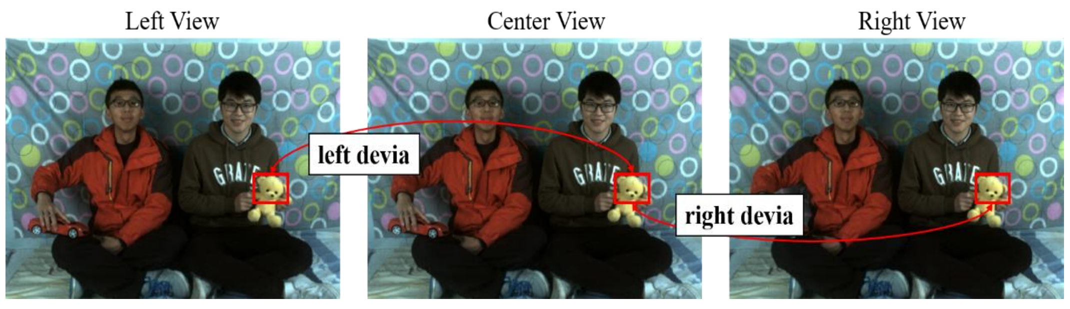


Fig.12 Disparity position of same object

As is shown in Fig.12, because the time interval between two frames is very short, the deviation in frame 2 is assumed to be equal to the deviation in frame 1 by default.[4] In frame 1, we take a scanning tape whose pixel number is one third of pixel number of the whole image. Respectively, the reference tape of the center image is steadfast, while the scanning tape of the surrounding image is moving in the opposite direction of deviation as shown in Fig.13. At each scanning step, we calculate the difference matrix between scanning tape and reference tape, as well as the sum of the absolute values of the elements in each difference matrix. The step with minimum sum is assumed to be the deviation in certain direction.

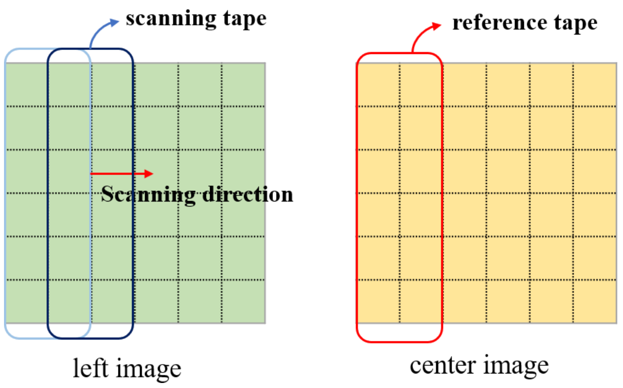


Fig. 13 Scanning for deviation

The deviations obtained from above method are used to correct the surrounding images. As for correction of the left image, suppose the left deviation is *a, col* and *row* are column and row length of the image, we pick ($a$ + 1 : $col$) columns of left image and ($col$ – $a$ + 1 : $col$) columns of center image to compose the new *left correction image*. The correction of other surrounding images is alike. Note that while correcting images in frame 2, we cannot use the known center image, so we replace it with center image of frame 1.[5] The idea of the correction procedure is shown below in Fig.14.

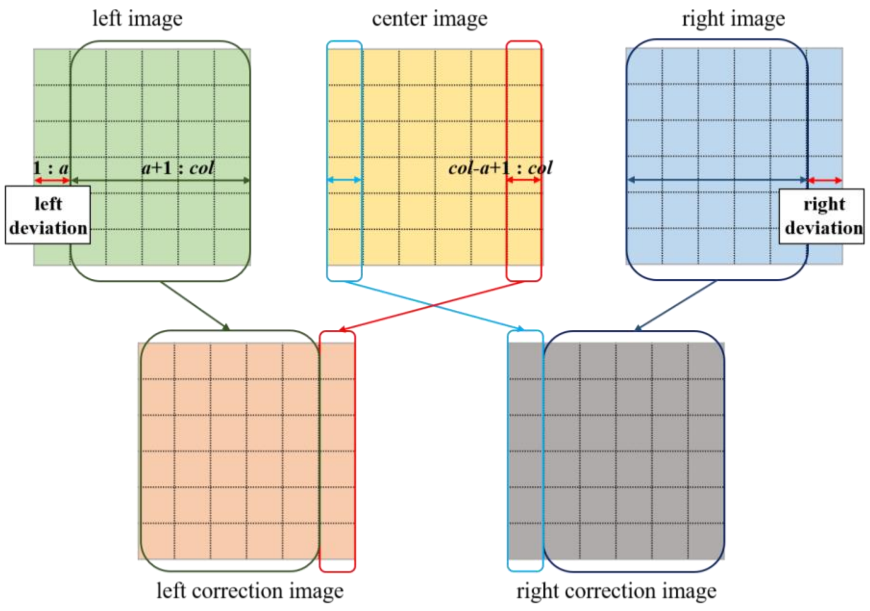


Fig.14 Correction principle of left and right images

### 2.2.3 Object Segmentation

The basic model only takes spatial differences into account and it will surely show worse performance on datasets with some objects changing a lot. Hence, we should improve the model by taking temporal differences into account.

By contrasting surrounding images in frame 1 with corresponding images in frame 2, we can find the objects that changed a lot within two frames. Some examples are shown before in Fig.7 to Fig.9. Then we need to extract these changing parts from the unchanged backgrounds.

The aim of segmentation is to extract changing parts from the difference images *img_xx_diff_C* (named according same law mentioned before). Note that these difference images are the differences between images captured by same camera in frame 1 and 2. This is the temporal difference, not like the spatial difference defined in section 2.2.1 (differences within frame 1). That is to say,

$$\begin{aligned} img1\_xx\_diff\_C &= img1\_mc\_C - img1\_xx\_C \\ img\_xx\_diff\_C &= img2\_xx\_C - img1\_xx\_C \end{aligned} \tag{3}$$

where *xx* can be *ul*, *uc*, *ur*, *ml*, *mr*, *dl*, *dc*, *dr*.

To segment changing object, we need to create an *Index matrix* with same size of the difference images. They we use the block to do traversal in the difference images (after thresholding), if there the number of 1s is larger than certain pre-set number, then we determine the current block as the changing part and set corresponding elements in *Index matrix* all 1s, and the elements of the *Index matrix* corresponding to unchanged parts are set to 0s. The final segmentation result is the union of *R, G, B* channels. Here are some examples of segmentation.

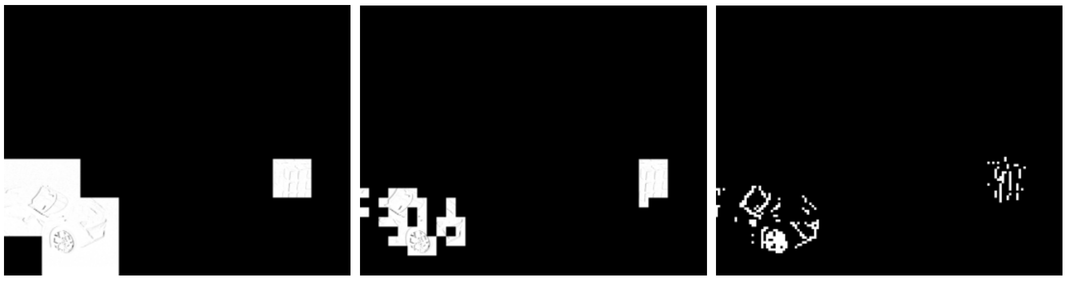


Fig.15 Segmentation for Data/Toys (*block_size* = 64, 16, 4)

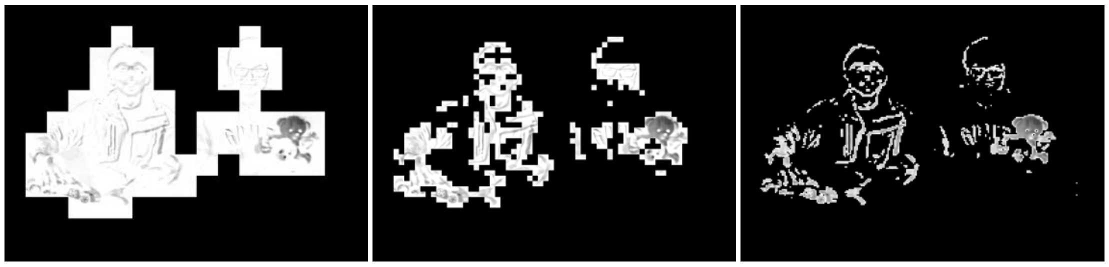


Fig.16 Segmentation for Data/Boys (*block_size* = 64, 16, 4)

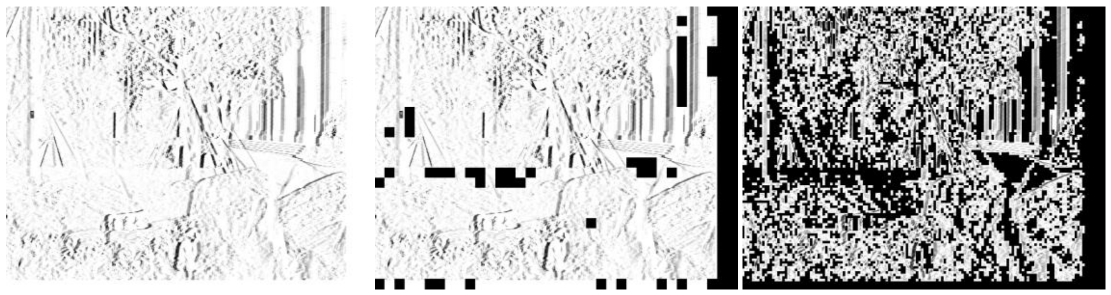


Fig.17 Segmentation for Data/Trees (*block_size* = 64, 16, 4)

From Fig.15 and Fig.17 we can see that our algorithm can successfully segment the changing parts from the unchanged backgrounds. This will absolutely do benefit to the improvement of our model.

However, in Fig.17, we can see that when we set *block_size* to 64×64, the whole image is assumed as changing part, which implies that Data/Trees has different changing pattern from other datasets. After precise checking we find that cameras moved right in some scale between two frames in Data/Trees. This shed light upon us that even though the segmentation might improve model performance on other datasets, it is not enough for Data/Trees. We need further approaches to deal with Data/Trees.

### 2.2.4 Improved model (Spatial & Temporal Difference Model)

Integrating deviation correction and object segmentation, we can get 9 corrected and segmented images of *R, G, B* channels. Then we will optimize the changing parts and unchanged parts separately.

As for changing parts, we multiply the up, down, left, right images in frame 2 with the *Index matrix* as references in optimization. The reason why we do not use images in frame 1 is that the differences between two frames are too large and will degenerate the quality of predicted images. The optimization model for changing parts is:

$$\begin{aligned} \text{minimize} \quad & \sum_{n=1}^{4} \| X_{pred} - Ref_n \|_F \\ \text{subject to} \quad & X(i,j) \geq 0; \\ & X(i,j) \leq 255; \\ & Ref_n = img2\_xx\_devia \cdot Index; \\ & i = 1,2,\dots,block\_size; \\ & j = 1,2,\dots,block\_size; \\ & xx = ul, ml, mr, dc. \end{aligned} \tag{4}$$

Still, this model is implemented block-wise and in three channels separately. Note that in the objective function there is only spatial difference constraints.

As for unchanged parts, we multiply the same four images with (1 - *Index*). Here we not only use images in frame 2 but also use the differences between frame 1 and frame 2 as references. If we only use images of frame 1 or frame 2, there would be some information lost. The first term can be interpreted as the difference generated by light field and other environmental factors. Hence, we add these surrounding differences respectively to the center image as the main source of information to predict the center image in frame 2. The second term is supplementary term for amelioration of texture and detailed information. (Fig. 18).

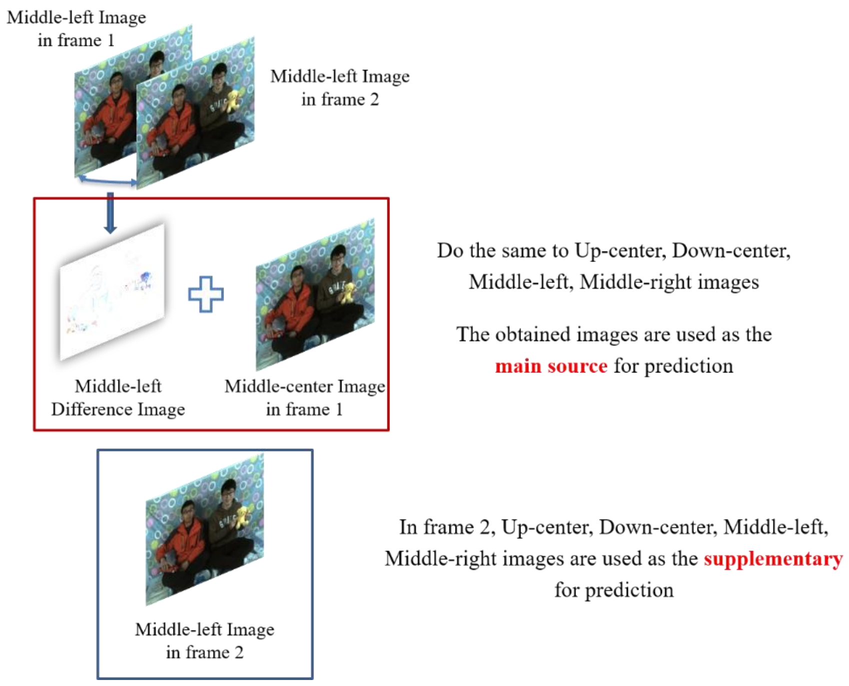

Middle-left Image
in frame 1
Middle-left Image
in frame 2
Middle-left
Difference Image
Middle-center Image
in frame 1
Do the same to Up-center, Down-center,
Middle-left, Middle-right images
The obtained images are used as the
main source for prediction
Middle-left Image
in frame 2
In frame 2, Up-center, Down-center, Middle-left,
Middle-right images are used as the supplementary
for prediction

Fig.18 Illustration of the objective function (unchanged part)

The model for unchanged part is:

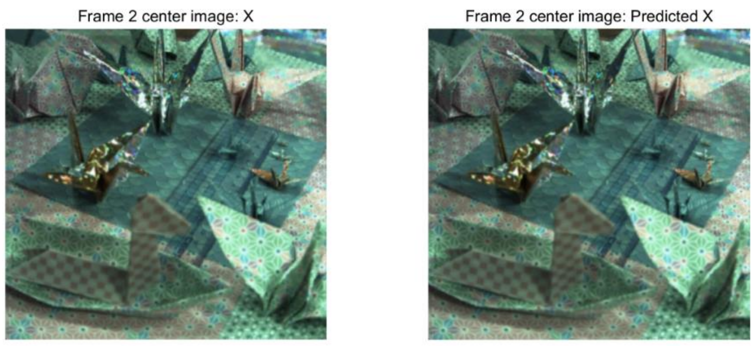
$$
\begin{aligned}
\text{minimize} \quad & \sum_{n=1}^{4} \| X_{pred} - Ref_n \|_F + 0.5 \cdot \sum_{m=1}^{4} \| X_{pred} - img2\_xx\_ref \|_F \\
\text{subject to} \quad & X(i,j) \ge 0; \\
& X(i,j) \le 255; \\
& Ref_n = (img1\_mc\_devia + img\_xx\_diff\_devia) \cdot (\mathbf{1} - Index); \\
& img\_xx\_diff\_devia = img2\_xx\_devia - img1\_xx\_devia; \\
& img2\_xx\_ref = img2\_xx\_devia \cdot (\mathbf{1} - Index) \\
& i = 1, 2, \ldots, block\_size \\
& j = 1, 2, \ldots, block\_size; \\
& xx = ul, ml, mr, dc.
\end{aligned}
\qquad (5)
$$

Note that the first term of the objective function is temporal difference and the second term is spatial difference.

## III. Model Evaluation and Results Analysis

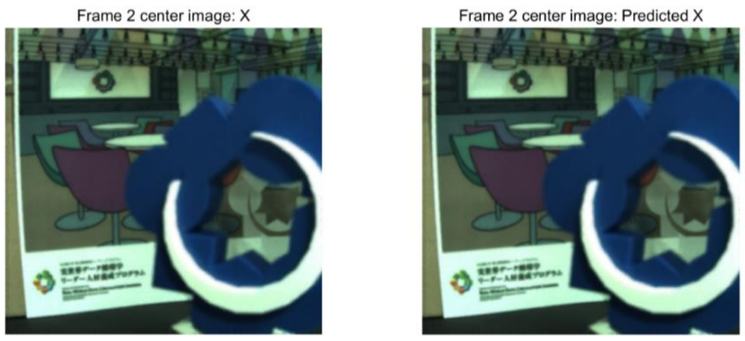
### 3.1 Evaluation and analysis of basic model

Implement our basic model (Spatial Difference Model) in given 7 datasets, we can get the predicted image and corresponding PSNR by comparing them with true center images X. Here we set *block_size* as 64×64 and equal weights.



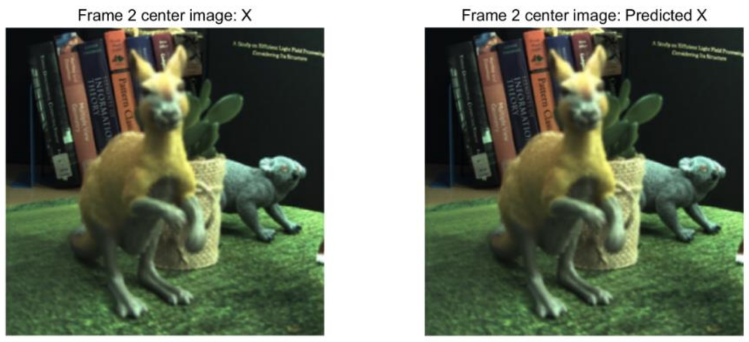
```
Overall Running Time: 466.7007s
PSNR of predicted Data/NagoyaOrigami center image: 49.2185
```

Fig.19 Comparison and PSNR of Data/NagoyaOrigami



```
Overall Running Time: 597.722961s
PSNR of predicted Data/NagoyaDataLeading center image: 50.1268
```

Fig.20 Comparison and PSNR of Data/NagoyaDataLeading



```
Overall Running Time: 566.619224s
PSNR of predicted Data/NagoyaFujita center image: 49.6233
```

Fig.21 Comparison and PSNR of Data/NagoyaFujita

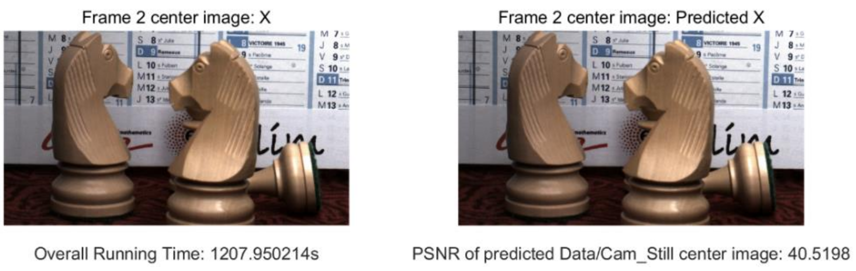

Frame 2 center image: X
Frame 2 center image: Predicted X
Overall Running Time: 1207.950214s
PSNR of predicted Data/Cam_Still center image: 40.5198

Fig.22 Comparison and PSNR of Data/Cam_Still

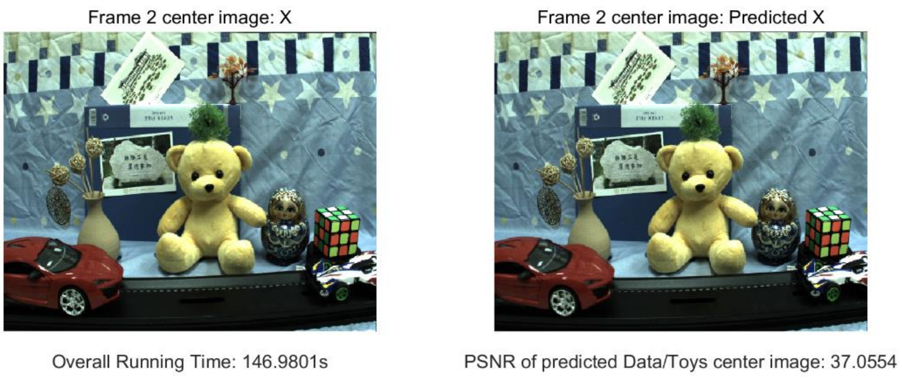

Frame 2 center image: X
Frame 2 center image: Predicted X
Overall Running Time: 146.9801s
PSNR of predicted Data/Toys center image: 37.0554

Fig.23 Comparison and PSNR of Data/Toys

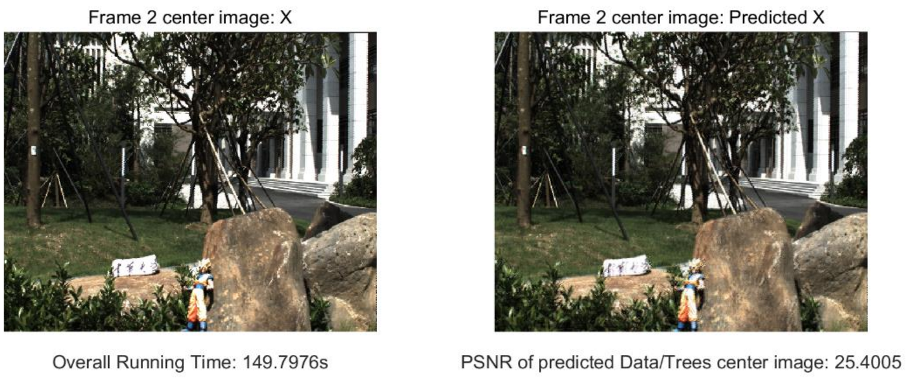

Frame 2 center image: X
Frame 2 center image: Predicted X
Overall Running Time: 149.7976s
PSNR of predicted Data/Trees center image: 25.4005

Fig.24 Comparison and PSNR of Data/Trees

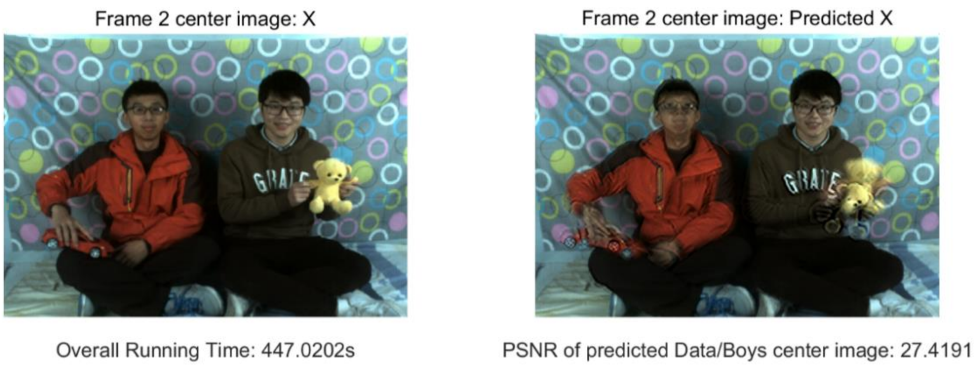

Frame 2 center image: X
Frame 2 center image: Predicted X
Overall Running Time: 447.0202s
PSNR of predicted Data/Boys center image: 27.4191

Fig.25 Comparison and PSNR of Data/Boys

From Fig.19 to Fig.25 we can see the results are just as we analyzed before in section 2.2.2. The predicted images of the first 3 datasets (Nothing changed greatly) show quite high PSNR values: 49.2185, 50.1268, 49.6233 respectively.

On contrary, the predicted images of rest 4 datasets show degenerated performance with PSNR values of 40.48, 37.0554, 25.3457, 27.4191 respectively. The mean PSNR of 7 datasets is 39.9012. The more the objects in the scene moved, the worse performance the basic model shows. Indeed, we can easily find that there are ghosts in the whole scene for Data/Trees and in the yellow toy bear and red toy car for Data/Boys. These are just the segmented objects we found in section 2.2.2.

The block numbers are 210, 210, 210, 255, 63, 63, 204 respectively.

We can draw a brief conclusion of our basic model that it is only suitable for prediction in less-changing scenes and objects and will unavoidably show worse performance in images with large-scale moving objects.

## 3.2 Evaluation and analysis of improved model

Implement our improved model (Spatial & Temporal Difference Model) in given 7 datasets, we can get the predicted image and corresponding PSNR by comparing them with true center images X. Here we set *block_size* as 64×64 and 16×16.

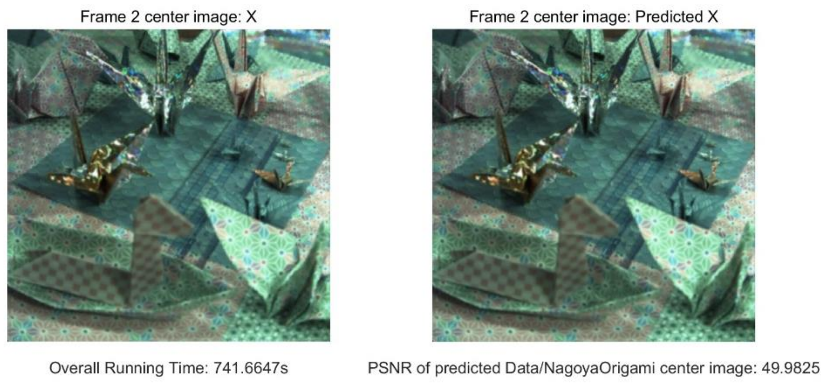

Frame 2 center image: X
Frame 2 center image: Predicted X
Overall Running Time: 741.6647s
PSNR of predicted Data/NagoyaOrigami center image: 49.9825

Fig.26 Comparison and PSNR of Data/NagoyaOrigami (*block_size* = 64)

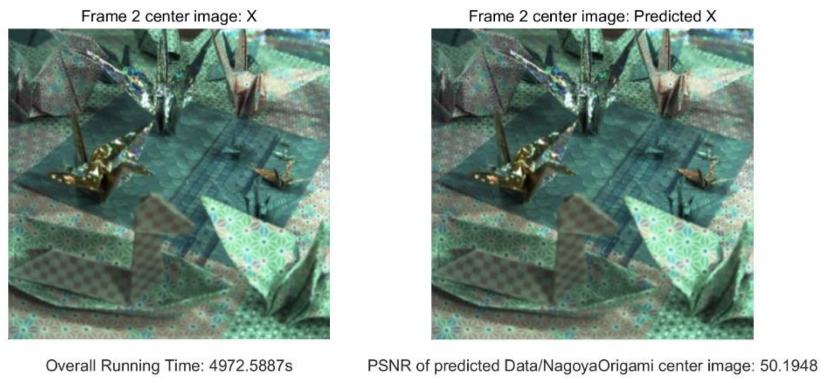
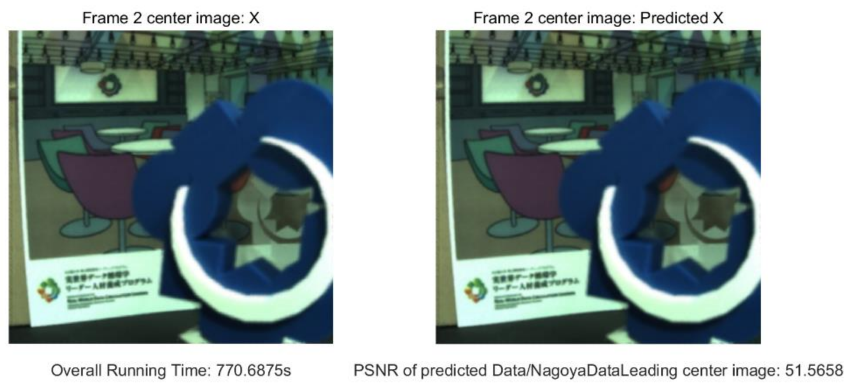
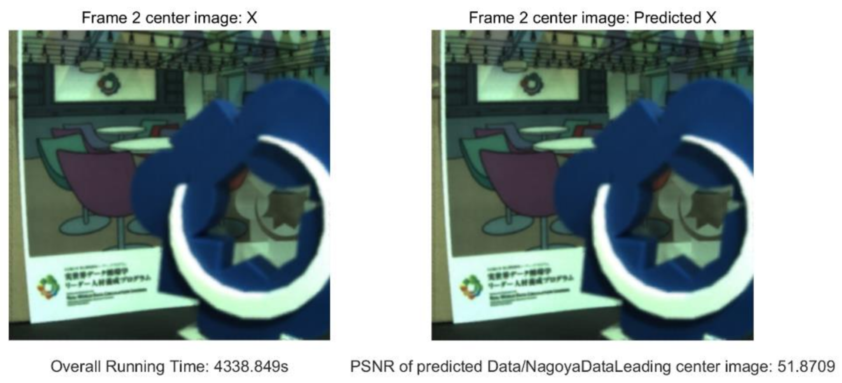

Frame 2 center image: X
Frame 2 center image: Predicted X
Overall Running Time: 4972.5887s
PSNR of predicted Data/NagoyaOrigami center image: 50.1948

Fig.27 Comparison and PSNR of Data/NagoyaOrigami (*block_size* = 16)



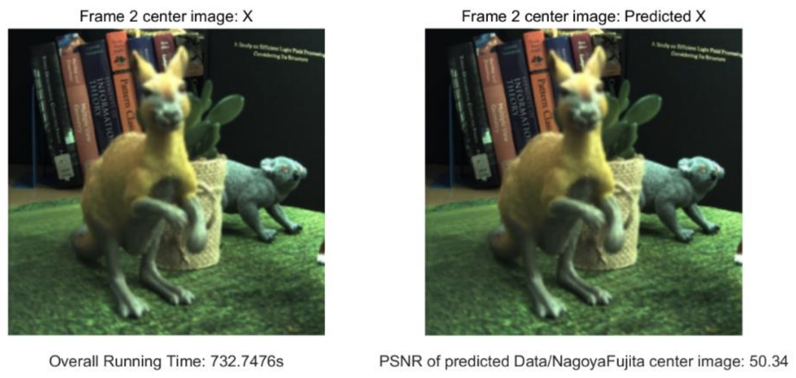
Overall Running Time: 770.6875s　PSNR of predicted Data/NagoyaDataLeading center image: 51.5658

Fig.28 Comparison and PSNR of Data/NagoyaDataLeading (*block_size* = 64)



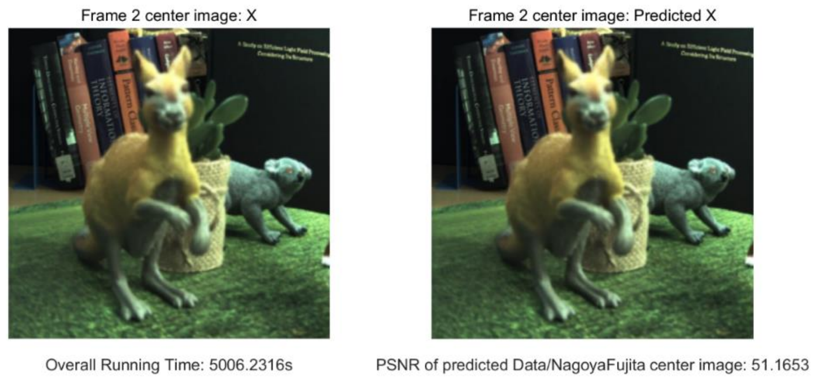
Overall Running Time: 4338.849s　PSNR of predicted Data/NagoyaDataLeading center image: 51.8709

Fig.29 Comparison and PSNR of Data/NagoyaDataLeading (*block_size* = 16)



Overall Running Time: 732.7476s　PSNR of predicted Data/NagoyaFujita center image: 50.34

Fig.30 Comparison and PSNR of Data/NagoyaFujita (*block_size* = 64)



Overall Running Time: 5006.2316s　PSNR of predicted Data/NagoyaFujita center image: 51.1653

Fig.31 Comparison and PSNR of Data/NagoyaFujita (*block_size* = 16)

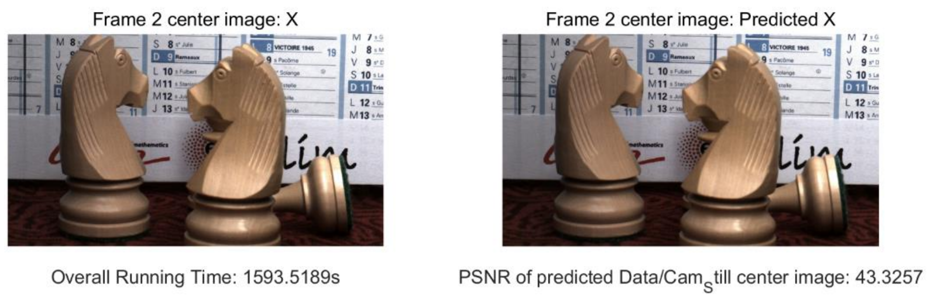

Frame 2 center image: X
Frame 2 center image: Predicted X
Overall Running Time: 1593.5189s
PSNR of predicted Data/Cam$_S$till center image: 43.3257

Fig.32 Comparison and PSNR of Data/Cam_Still *(block_size* = 64)

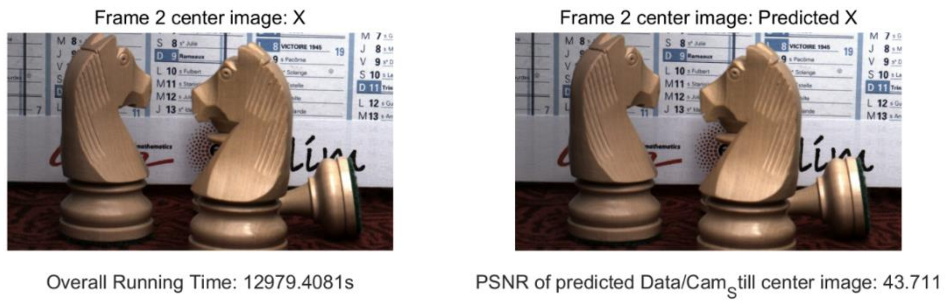

Frame 2 center image: X
Frame 2 center image: Predicted X
Overall Running Time: 12979.4081s
PSNR of predicted Data/Cam$_S$till center image: 43.711

Fig.33 Comparison and PSNR of Data/Cam_Still *(block_size* = 16)

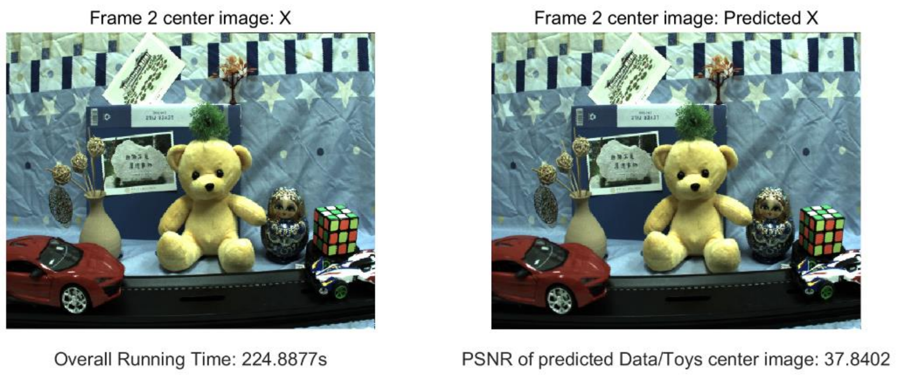

Frame 2 center image: X
Frame 2 center image: Predicted X
Overall Running Time: 224.8877s
PSNR of predicted Data/Toys center image: 37.8402

Fig.34 Comparison and PSNR of Data/Toys *(block_size* = 64)

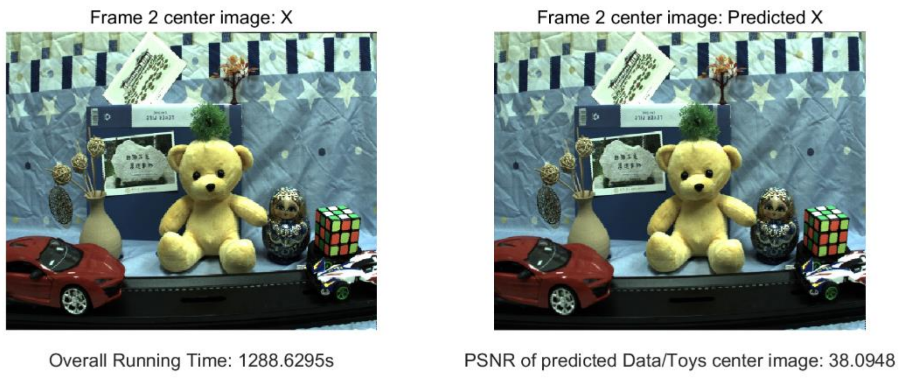
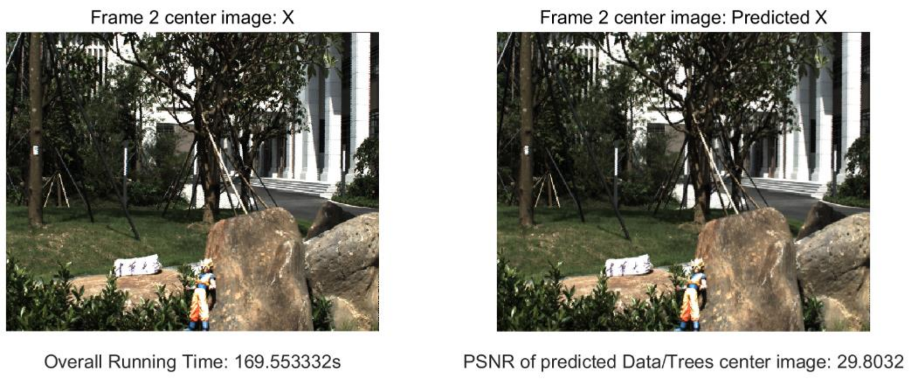

Frame 2 center image: X
Frame 2 center image: Predicted X
Overall Running Time: 1288.6295s
PSNR of predicted Data/Toys center image: 38.0948

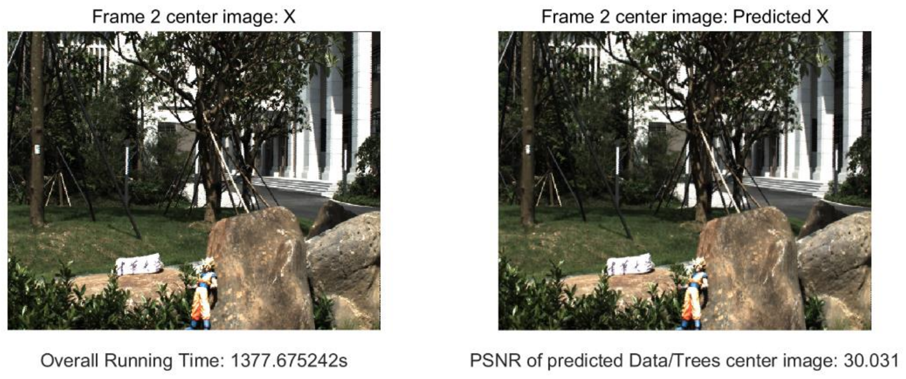
Fig.35 Comparison and PSNR of Data/Toys *(block_size* = 16)



Overall Running Time: 169.553332s

PSNR of predicted Data/Trees center image: 29.8032

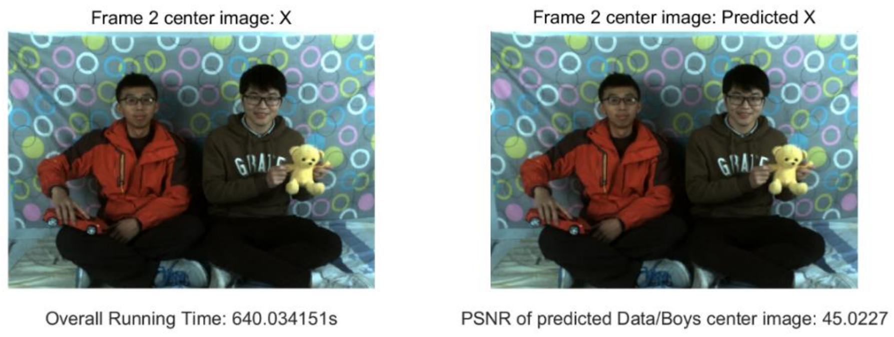
Fig.36 Comparison and PSNR of Data/Trees *(block_size* = 64)



Overall Running Time: 1377.675242s

PSNR of predicted Data/Trees center image: 30.031

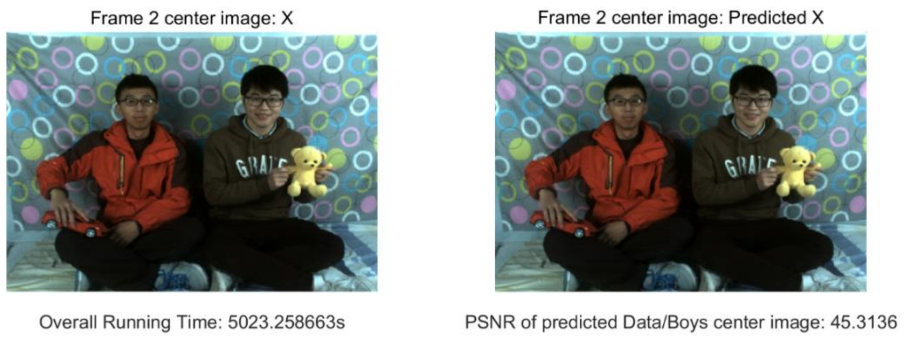
Fig.37 Comparison and PSNR of Data/Trees *(block_size* = 16)



Overall Running Time: 640.034151s

PSNR of predicted Data/Boys center image: 45.0227

Fig.38 Comparison and PSNR of Data/Boys *(block_size* = 64)



Overall Running Time: 5023.258663s

PSNR of predicted Data/Boys center image: 45.3136

Fig.39 Comparison and PSNR of Data/Boys *(block_size* = 16)

As we can see from Fig.26 to Fig. 39, our improved model shows significant improvement in images with greatly changing objects as well as enhances the prediction effects of texture and light field information of the background.

The block numbers are 210, 210, 210, 255, 63, 63, 204 respectively with *block_size* of 64 and 3135, 3135, 3135, 8160, 952, 952, 3149 respectively with *block_size* of 16.

Table.2 Comparison of PSNR

| | Basic model | Improved model | |
|---|---|---|---|
| *block_size* | 64*64 | 64*64 | 16*16 |
| NagoyaOrigami | 49.2185 | 49.9825 | 50.1948 |
| NagoyaDataLeading | 50.1268 | 51.5658 | 51.8709 |
| NagoyaFujita | 49.6233 | 50.3400 | 51.1653 |
| Cam_Still | 40.5198 | 43.3257 | 43.7110 |
| Toys | 37.0554 | 37.8402 | 38.0948 |
| Trees | 25.4005 | 29.8032 | 30.0310 |
| Boys | 27.4191 | 45.0227 | 45.3136 |
| Average | 39.9091 | 43.9829 | 44.3402 |

Table.3 Comparison of optimization time

| | Basic model | Improved model | |
|---|---|---|---|
| *block_size* | 64*64 | 64*64 | 16*16 |
| NagoyaOrigami | 467s | 742s | 4973s |
| NagoyaDataLeading | 598s | 741s | 4339s |
| NagoyaFujita | 567s | 733s | 5006s |
| Cam_Still | 1208s | 1594s | 12979s |
| Toys | 147s | 225s | 1289s |
| Trees | 150s | 170s | 1378s |
| Boys | 447s | 640s | 5023s |

From Table.2 we can see that the improved model prevails over basic model in all datasets. As we have discussed in last section, our improved model utilizes both spatial difference information and temporal difference information, it also draws attention on the underlying environmental difference information.

After code optimization, the efficiency of our model also improved and the overall running time does not increase too much. The comparison of time is shown in Table.3

## IV. Conclusions and Further discussion

In this project, we analyzed the characteristics of given 7 groups of datasets and proposed a Spatial Difference Model (SDM) which only focuses on the differences between images in frame 1 based on the assumption that everything does not change too much between two frames.

After analyzing the pros and cons of SDM, we proposed two further processing improvements: deviation correction and object segmentation. Based on these two improvements, we proposed a Spatial & Temporal Model (STDM) which utilizes both spatial and temporal difference information and we believe it can extract the light field and other environmental information in some extent thus enhance the final performance.

Yet this is not the end. We also made some other improvement attempts such as adaptive weights in the objective function. Besides, since the PSNR of Data/Trees is the worst, we tried to implement the camera offset correction but it shows PSNR of 29.0369, worse than STDM.

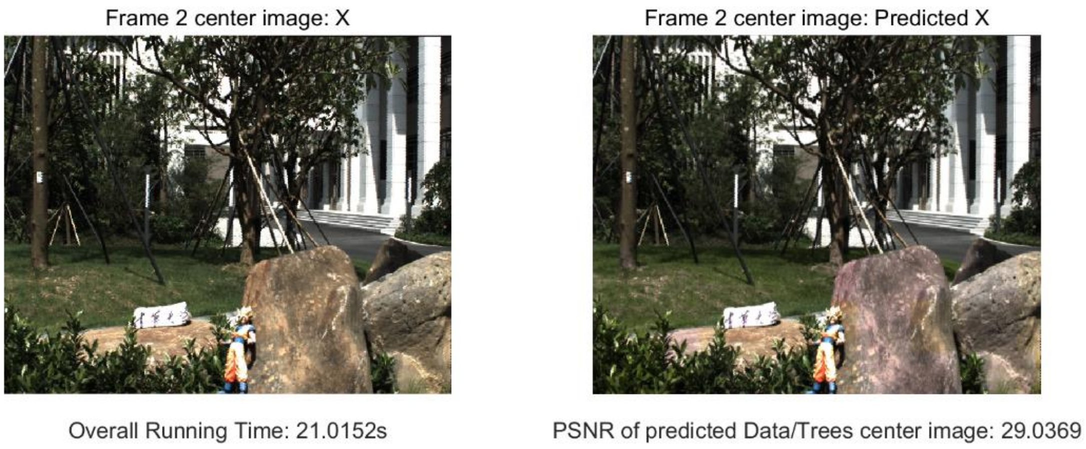

Frame 2 center image: X
Frame 2 center image: Predicted X
Overall Running Time: 21.0152s
PSNR of predicted Data/Trees center image: 29.0369

Fig. 40 Evaluation of camera offset correction

We can see that in the right image of Fig. 40, stone shows different color from the real stone in left image, the true center image in frame 2. We think it is because our method of camera offset correction is not rigorous enough to explain for the difference in Data/Trees, let alone there is much depth information in this dataset.

Therefore, we could try to solve the camera offset, rotation and other transformation and resulting changes in images and propose more effective and explanatory model in the future.

## References

[1] Lin J P, Tang A C W. A fast direction predictor of inter frame prediction for multi-view video coding[C]//2009 IEEE International Symposium on Circuits and Systems. IEEE, 2009: 2589-2592

[2] Avramelos V, Van Wallendael G, Lambert P. Overview of MV-HEVC prediction structures for light field video[C]//Applications of Digital Image Processing XLII. International Society for Optics and Photonics, 2019, 11137: 111371F.

[3] Li X, Zhao D, Ji X, et al. A fast inter frame prediction algorithm for multi-view video coding[C]//2007 IEEE International Conference on Image Processing. IEEE, 2007, 3: III-417-III-420.

[4] Ding L F, Tsung P K, Chien S Y, et al. Content-aware prediction algorithm with inter-view mode decision for multiview video coding[J]. IEEE Transactions on Multimedia, 2008, 10(8): 1553-1564.

[5] Schwarz H, Wiegand T. Inter-view prediction of motion data in multiview video coding[C]//2012 Picture Coding Symposium. IEEE, 2012: 101-104.