

# 凸优化实验报告

Inter-Prediction for Multiview Video Coding

专业: 自动化系 控制工程

姓名: Qiranzou, Ruonan Jia

2019 年 12 月 20 日

# 多视点视频编码的帧间预测

## Inter-Prediction for Multiview Video Coding

### 摘要 :

本文提出了微位移方法用来减小最优集的范围，使模型的预测结果更加稳定，减少了预测结果中的噪声；提出了 weight 与 transformer 两种模型，分别从权重角度和视角变换两个思路设计变换矩阵，并通过凸优化方法来求解变换矩阵。结合了微位移方法的两种模型在不使用其他图像处理方法的情况下，仅通过凸优化方法就获得了极高的 PSNR 值与视觉效果良好的预测图片。

报告将详细介绍微位移方法、weight 模型和 transformer 模型以及这些方法的前提假设，并阐述实验过程中遇见的问题以及模型设计历程中提出的试验性模型。报告的模型结果部分将展示每个模型预测的 PSNR 值和图片，对比实验部分将对比证明微位移方法的有效性、transformer 模型使用当前帧信息解决上一帧视差信息不足的可行性，并对比不同模型参数、不同模型结构对预测结果的影响。

本文采用 matlab 自带的 psnr 函数计算各数据集预测结果的指标，transformer 模型最优阈值时的预测结果如下：

Dataset	Boys	Cam_still	Leading	Fujita	Origami	Toys	Trees	Average
PSNR	42.2212	41.0174	52.6149	52.7888	49.8917	37.5560	28.2809	43.4816

关键字：微位移，weight 模型，transformer 模型，数据分析与可视化，PSNR

声明：若此报告内容所涉及方法理论对您的研究有所有帮助，我们将不胜荣幸，但我们也保留对此内容的所有权。若有需要，可与我们邮件联系：

[zouqr19@mails.tsinghua.edu.cn](mailto:zouqr19@mails.tsinghua.edu.cn)  
[jrn19@mails.tsinghua.edu.cn](mailto:jrn19@mails.tsinghua.edu.cn)

# 目录

<b>1 任务概述 .....</b>	<b>1</b>
1.1 任务背景 .....	1
1.2 任务内容 .....	1
1.3 任务要求 .....	2
1.4 评价指标 .....	2
<b>2 任务与数据分析 .....</b>	<b>3</b>
2.1 各数据集的 PSNR 矩阵数据分析 .....	3
2.2 各数据集的主观评价 .....	9
2.3 时域帧间分析 .....	11
2.4 空域帧间分析 .....	11
2.5 镜头分析 .....	11
<b>3 模型设计概述 .....</b>	<b>12</b>
3.1 创新点 .....	12
3.1.1 weight 模型概述 .....	12
3.1.2 transformer 模型概述 .....	12
3.1.3 微位移 .....	13
3.1.4 使用凸优化模型求得视差的变换 .....	15
3.2 前提与假设 .....	15
3.2.1 block 划分 .....	16
3.2.2 psnr 假设 .....	16
3.2.3 平移假设 .....	16
3.3 设计过程 .....	17
<b>4 模型细节 .....</b>	<b>19</b>
4.1 简单权重模型 .....	19
4.2 矩阵平移变换模型 v1 .....	21
4.3 矩阵平移变换模型 v2 .....	22
4.4 WEIGHT 模型 (较优) .....	23
4.5 帧间时域变化模型 (试验) .....	24
4.6 TRANSFORMER 模型 (最优) .....	26
<b>5 模型结果 .....</b>	<b>30</b>
5.1 TRANSFORMER 模型 .....	30
5.2 WEIGHT 模型 .....	32
5.3 其他模型 .....	34
5.3.1 简单权重模型 .....	34
5.3.2 矩阵平移变换模型 v1 .....	35
5.3.3 矩阵平移变换模型 v2 .....	36
5.3.4 帧间时域变化模型 .....	37

<b>6 对比实验 .....</b>	<b>38</b>
6.1     TRANSFORMER 与 WEIGHT 结果比较 .....	38
6.2     微位移方法对比实验 .....	38
6.2.1    微位移对 transformer 模型的影响.....	38
6.2.2    微位移对 weight 模型的影响 .....	39
6.3     TRANSFORMER 块大小参数与微位移量的选取 .....	40
6.4     TRANSFORMER 有无使用当前帧对比实验 .....	42
6.5     PSNR 近似矩阵阈值对 TRANSFORMER 结果的影响 .....	43
6.6     TRANSFORMER 有无水平垂直视角联合的结果对比.....	43
<b>7 总结与展望 .....</b>	<b>45</b>

# 1 任务概述

## 1.1 任务背景

随着计算机图形学和计算机视觉技术的发展，多视点视频（MVV）越来越多地引起人们的普遍关注。同传统的单视点视频相比，多视点视频具有丰富的三维深度信息，然而，由于多视点视频是由位置固定的多个摄像透镜同时从不同位置的角度拍摄同一场景而获得的一组视频信号，其数据量随着透镜数目的增多而成倍增加。巨大的视频数据量对存储和传输提出了更高的要求，而多视点视频编码（MVC）就是对多视点视频数据的有效压缩。图 1.1 表示多视点相机透镜阵列，图 1.2 表示多视点视频随时间变化的成像数据。

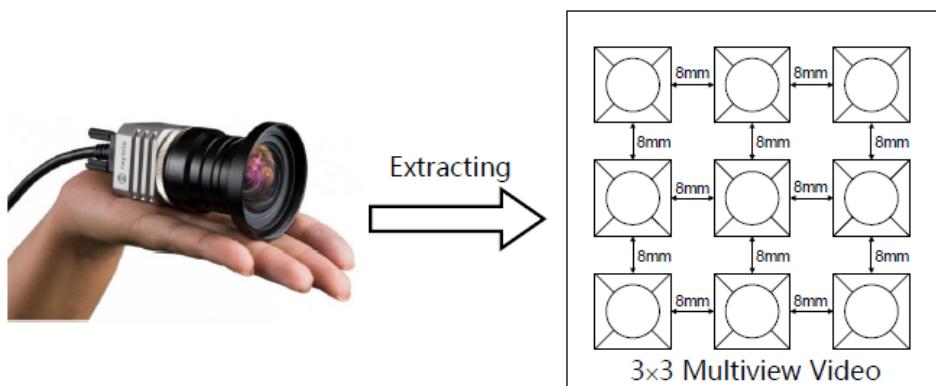


图 1.1 多视点相机透镜阵列

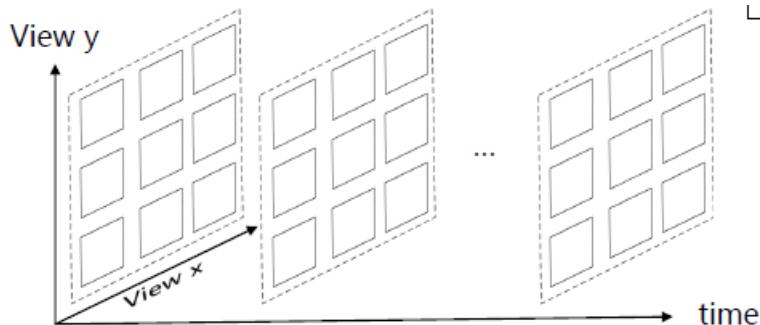


图 1.2 多视点视频随时间变化的成像数据

本课题是基于上述背景，利用多视点视频数据进行帧间预测，这些数据中除了具有同一视点的时间相关性以外，还具有同一时刻不同视点间的空间相关性。如何利用这些相关信息来进行预测是本次实验的关键所在。

## 1.2 任务内容

本次实验共给出 7 个数据集，分别为 Boys, Cam\_Still, NagoyaDataLeading, NagoyaFujita, NagoyaOrigami, Toys 和 Trees。每个数据集含有两帧视频内容，每帧内容又包括相机的 9 个透镜所拍到的图像信息。对两帧视频内容所含图像通过从左到右、从上到下的顺序进行编号，则第 1 帧 (Reference Frame) 图像内容为 R1-R9，第 2 帧 (Current Frame) 图像内容分别为 R10-R18。现将 R14 的图像从数据集中取出，单独编号为 X。图 1.3 表示视频帧的内容信息。

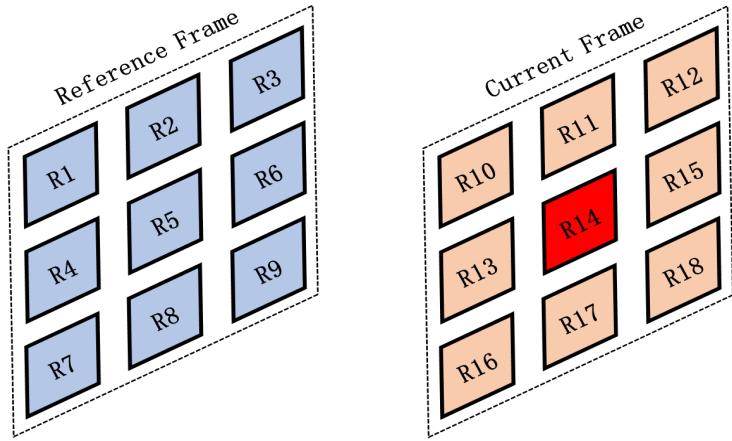


图 1.3 视频帧内容信息

我们所需解决的任务就是根据  $R_1, \dots, R_{13}$  和  $R_{15}, \dots, R_{18}$  这 17 张图像来预测  $R_{14}$  (编号为  $X'$ ) 的图像信息, 使之尽可能地与原图  $X$  相近。上述过程描述成数学问题就是 :

预测第 2 帧中间图像  $X'$ ,  $X' = f(R_1, \dots, R_{13}, R_{15}, \dots, R_{18})$

希望使得,  $\min Diff(X', X)$

### 1.3 任务要求

- (1) 原始图像  $X$  不能用于所建的优化模型 ;
- (2) 第 1 帧参考图像  $R_1-R_9$  和第 2 帧参考图像  $R_{10}-R_{13}$ 、 $R_{15}-R_{18}$ , 两者分别至少有一张图像用于模型中 ;
- (3) 预测图像时, 应该采取分块预测的方法 ;
- (4) 分块时, 应注意参考图像和预测图像中分块大小保持一致, 另外, 两者种分块的对应位置不一定相同。

### 1.4 评价指标

实验结果采取 PSNR 计算方式进行评估。PSNR 是“Peak Signal to Noise Ratio”的缩写, 即峰值信噪比, 是一种评价图像的客观标准, 它的计算方法如下 :

$$MSE = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W (X(i,j) - Y(i,j))^2$$

$$PSNR = 10 \times \log_{10} \left( \frac{(2^n - 1)^2}{MSE} \right)$$

其中,  $MSE$  表示当前图像  $X$  与参考图像  $Y$  的均方误差,  $H$ 、 $W$  分别为图像的高度和宽度,  $n$  为像素比特数。PSNR 的单位是 dB, 数值越大表示失真越小。

因为实验数据为彩色图像, 分为三个通道, 所以应对三个通道分别进行评价、求平均, 来作为评价结果 :

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

实验代码中直接采用 matlab 自带的 psnr 函数进行求解。

## 2 任务与数据分析

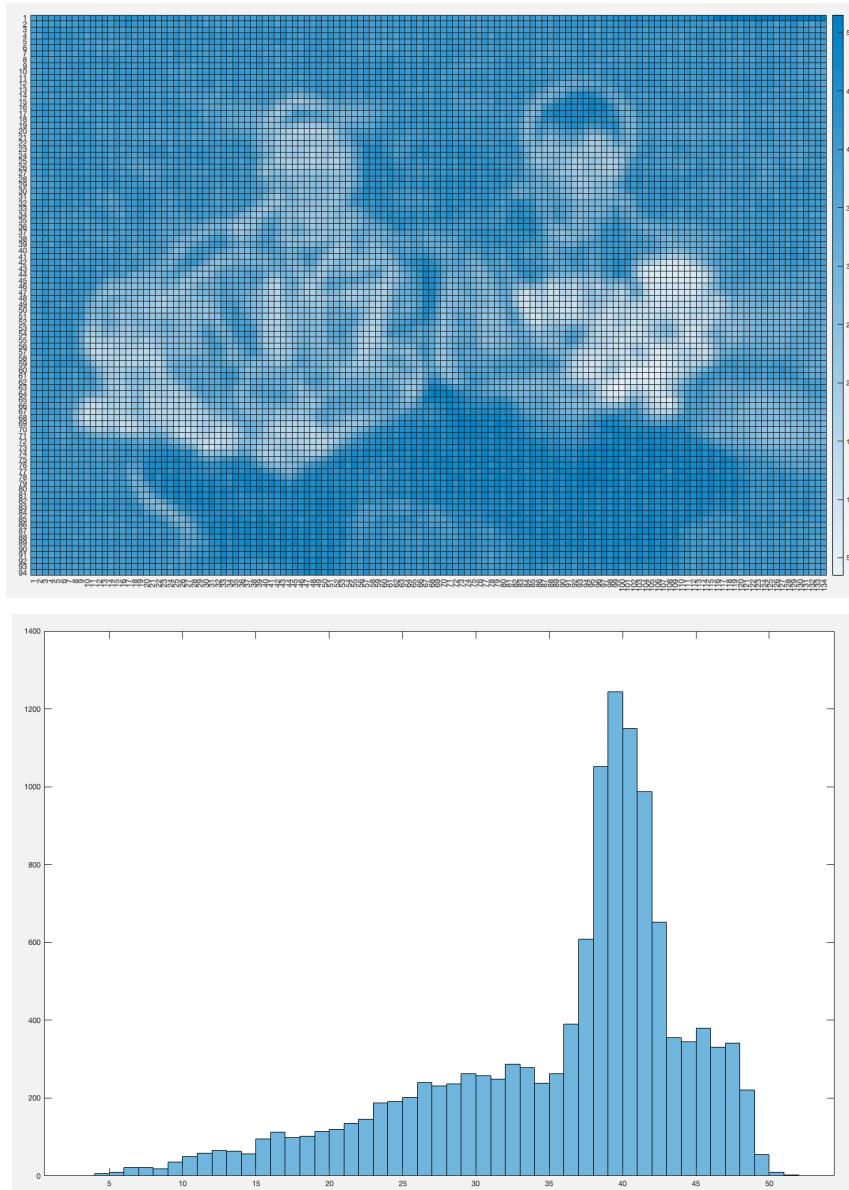
实验数据集共有 7 组，从 1-7 依次编号为 Boys, Cam\_Still, NagoyaDataLeading, NagoyaFujita, NagoyaOrigami, Toys 和 Trees。数据集中数据格式为 RGB 类型，像素比特数为 8，像素变化范围是[0, 255]。

### 2.1 各数据集的 psnr 矩阵数据分析

在各个数据集中，对参考图像进行分块处理，求出前后帧之间不同视角对应 block 图块的 psnr 平均值，从而得到各个数据集的 psnr 矩阵。然后从热力图和直方图角度，对不同数据集的 psnr 矩阵进行可视化分析。

(1) Boys 数据集：

该数据集 psnr 矩阵的热力图和直方图如下所示。



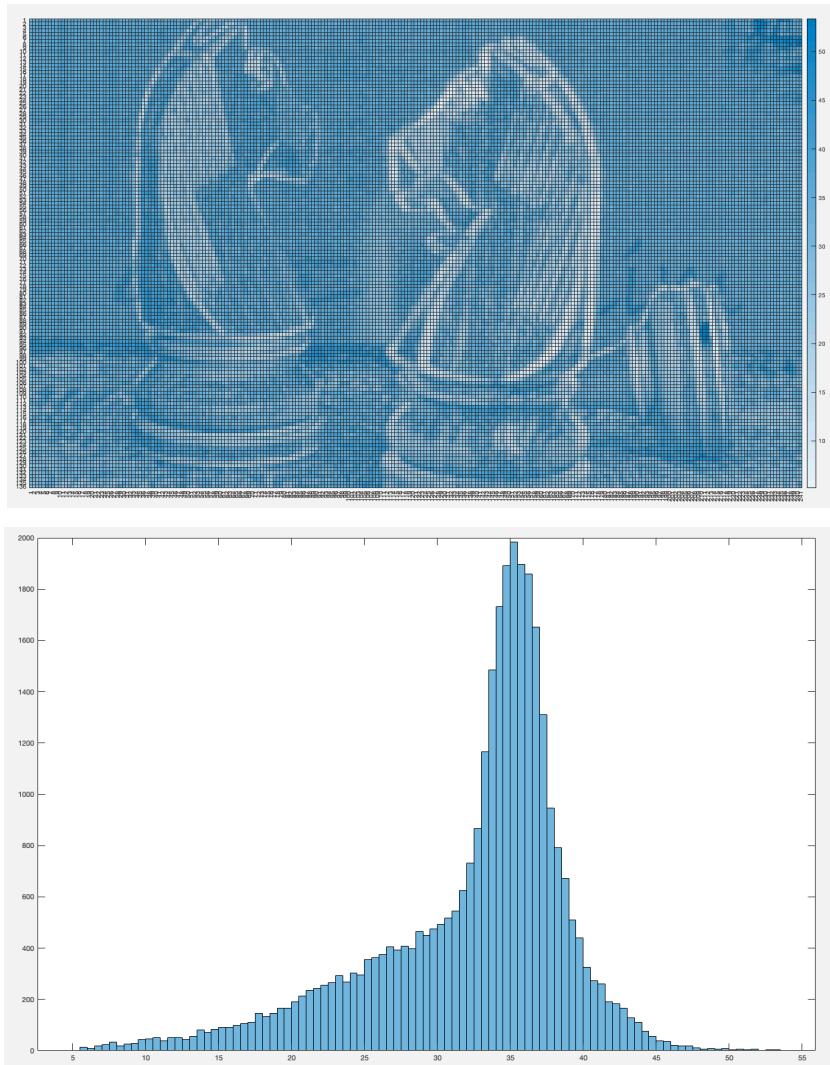
结果分析：

在 psnr 矩阵的热力图中，psnr 值较低的位置颜色较浅，表示在该位置的 block 图块前后帧之间差别较大；相反 psnr 值较高的位置颜色较深，表示该位置的 block 图块前后帧之间比较相近。因此，可以看出前后帧之间背景变化不大，整体变动体现在人物和玩偶身上。

另外，对 psnr 值进行直方图统计发现，图块中存在较多 psnr 值较低的部分，说明这个数据集还有很大的优化空间。

## (2) Cam\_Still 数据集：

该数据集 psnr 矩阵的热力图和直方图如下所示。

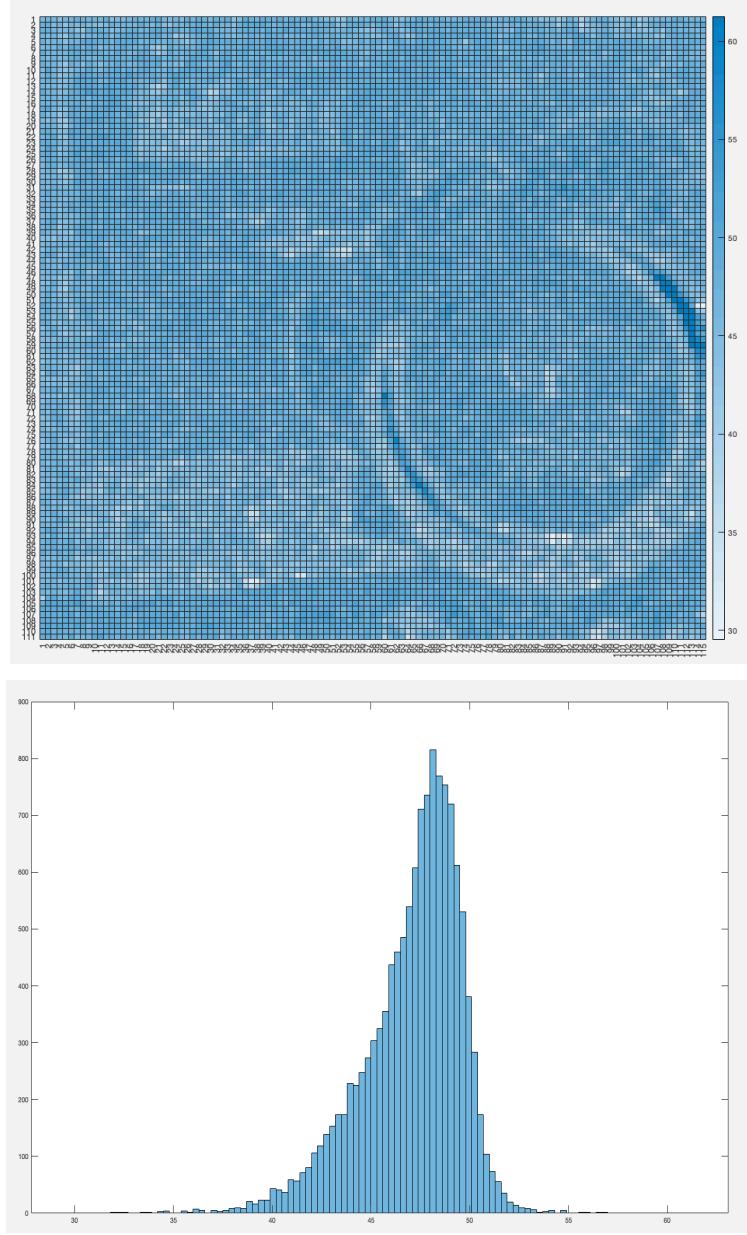


结果分析：

从 psnr 热力图中可以发现，木偶部分轮廓颜色较浅，因此前后帧之间变动主要体现在木偶身上；从 psnr 直方图中可以发现，整个 psnr 矩阵的平均值大约是 35，因此更加需要对这些均值以下的部分进行优化。

(3) NagoyaDataLeading 数据集：

该数据集 psnr 矩阵的热力图和直方图如下所示。

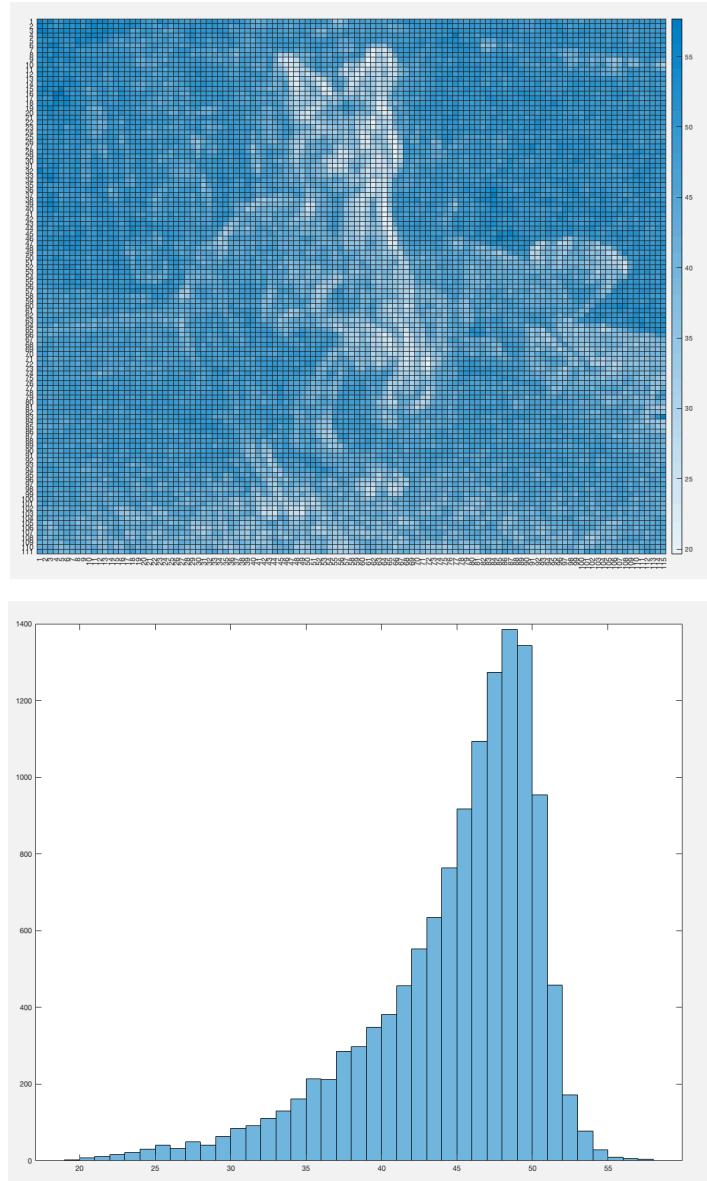


结果分析：

从 psnr 热力图中可以发现，整个图颜色较深，因此这个数据集前后帧之间变动较小；从 psnr 直方图中可以发现，整个 psnr 矩阵的值集中在 40 以上，更加直接说明前后帧之间的几乎没有出现变化。

(4) NagoyaFujita 数据集：

该数据集 psnr 矩阵的热力图和直方图如下所示。

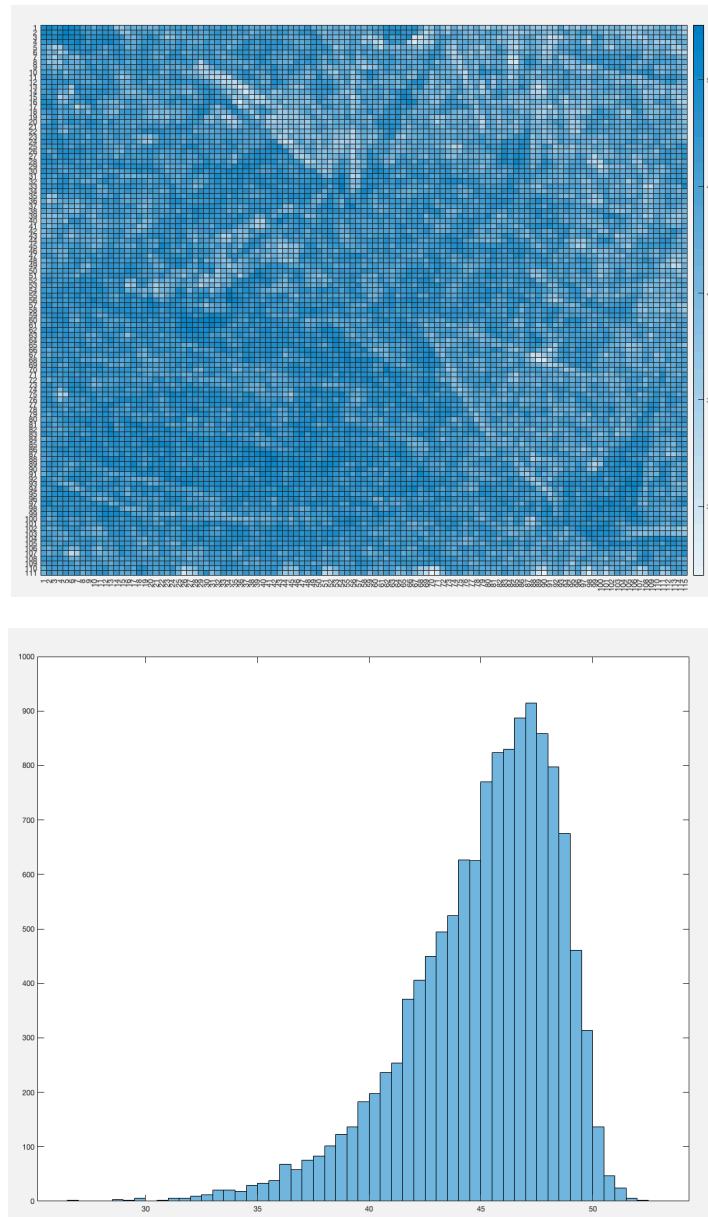


结果分析：

从 psnr 热力图中可以发现，袋鼠和浣熊头部、身体部位颜色较浅，因此前后帧之间变动主要体现在这两者上；从 psnr 直方图中可以发现，整个 psnr 矩阵的值多集中在 40-50 之间，也存在部分的值较低。

(5) NagoyaOrigami 数据集：

该数据集 psnr 矩阵的热力图和直方图如下所示。

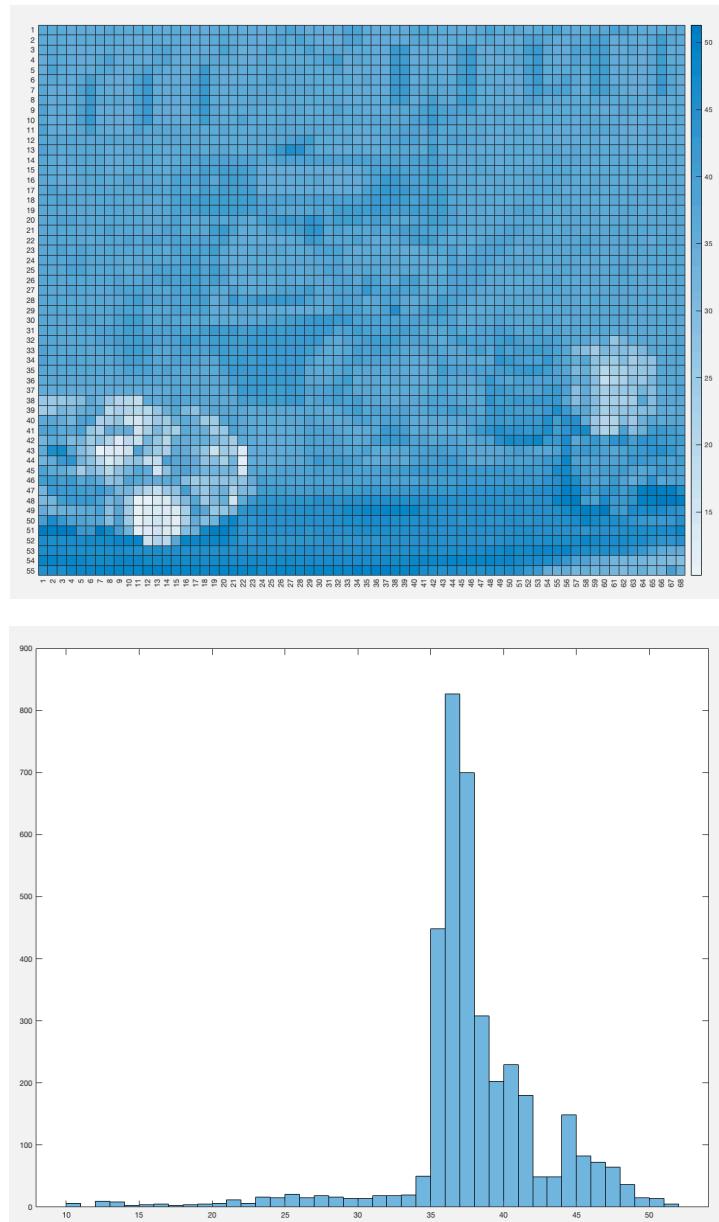


结果分析：

从 psnr 热力图中可以发现，整个图颜色较深，仅在纸鹤边缘部分出现浅色部分，说明前后帧之间在纸鹤处出现微小变动；从 psnr 直方图中可以发现，整个 psnr 矩阵的值也较多集中在 40-50 之间，几乎没有低于 30 的 block，前后帧之间整体匹配质量较高。

## (6) Toys 数据集：

该数据集 psnr 矩阵的热力图和直方图如下所示。

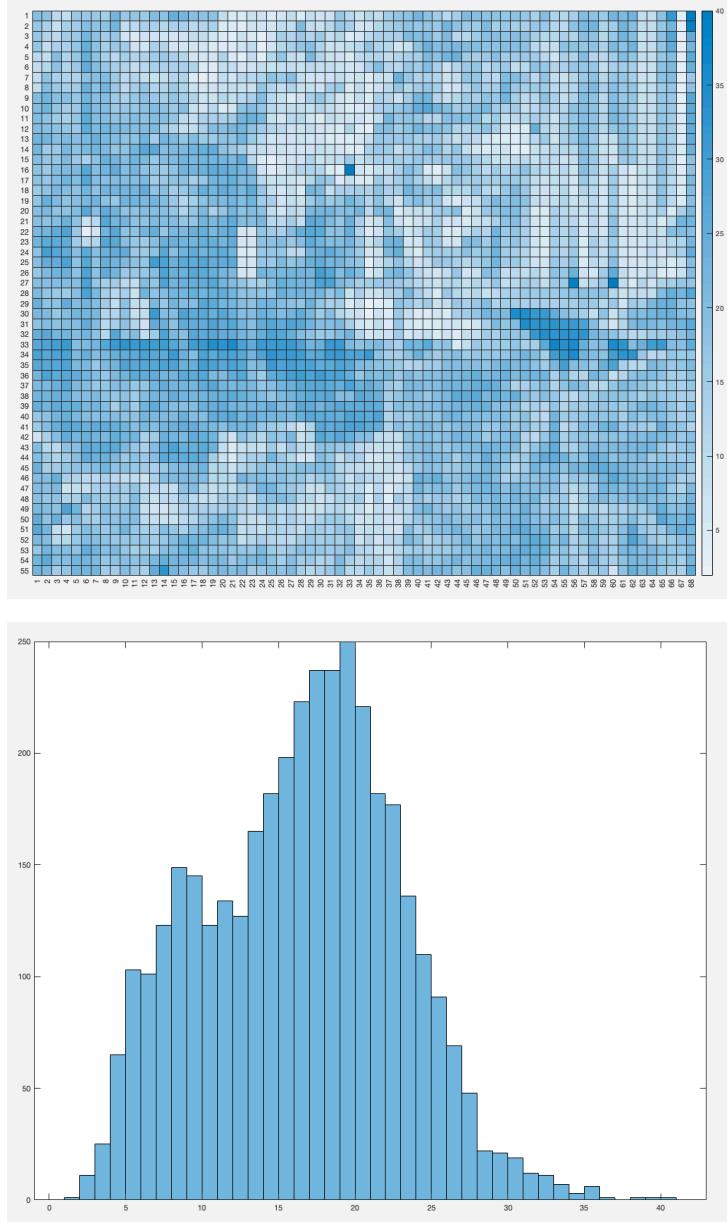


### 结果分析：

从 psnr 热力图中可以发现，左下角小车和右下角魔方的颜色较浅，因此前后帧之间变动主要体现在这两者上；从 psnr 直方图中可以发现，整个 psnr 矩阵的值多集中在 35 以上，并存在较少低值的 block 图块。

### (7) Trees 数据集：

该数据集 psnr 矩阵的热力图和直方图如下所示。



### 结果分析：

从 psnr 热力图中可以发现，整个图的颜色较浅，因此前后帧之间变化比较明显；从 psnr 直方图中可以发现，整个 psnr 矩阵的大部分值在 25 以下，因此前后帧之间对应 block 图块差别较大。

## 2.2 各数据集的主观评价

(1) Boys 数据集：时域上，背景几乎没有变化，前景中人物面部、手部动作和手中的玩具位置变化比较明显（如下图红框所示），对预测造成很大影响（挑战），与 2.1 (1) 分析结果相符；空域上，镜头阵列与拍摄物体平面基本保持一致，上下左右几乎处于平移关系。

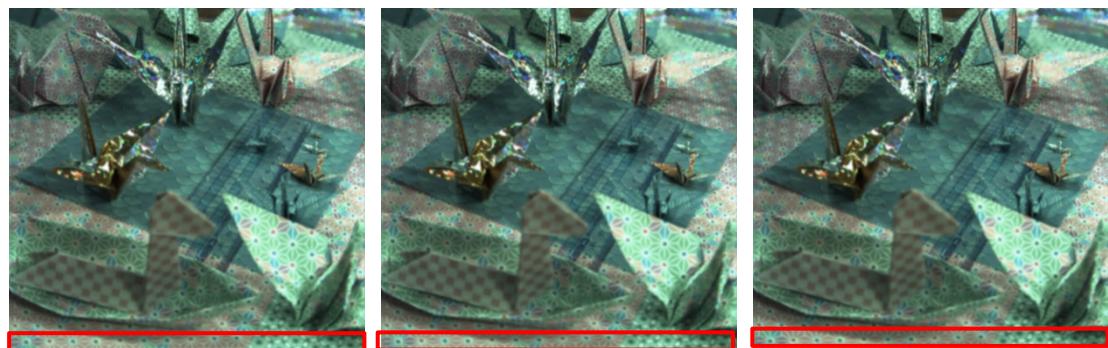


(2) Cam\_Still 数据集：时域上，墙面背景没有变化，桌面和上面的木偶发生旋转；空域上，摄像机不是正对物体，因此水平和竖直方向镜头所成的像发生一定程度的旋转偏移，其中水平方向的偏移较为明显

(3) NagoyaDataLeading 数据集：时域上，被拍摄物体几乎处于静止状态，前后帧之间几乎一致保持；空域上，镜头阵列与拍摄物体平面基本保持一致，上下视角、左右视角几乎处于平移关系。

(4) NagoyaFujita 数据集：时域上，背景保持不变，前景的袋鼠发生转动；空域上，镜头阵列与拍摄物体平面基本保持一致，左右和上下视角均处于平移状态。

(5) NagoyaOrigami 数据集：时域上，前后帧之间变化不明显；空域上，竖直方向镜头阵列所处的平面与被拍摄物体平面不平行，上中下视角拍摄物体变化幅度不一致，在下图红框出比较明显。



(6) Toys 数据集：时域上，图像内容主要是左下角的小车和右下角的魔方位置发生变化；空域上，镜头阵列与拍摄物体平面保持一致，视角变化比较均匀。

(7) Trees 数据集：时域上，前后帧之间存在明显的平移关系，在下图红框处较为明显；空域上，镜头阵列与拍摄物体平面基本保持一致，视角变化比较均匀。



### 2.3 时域帧间分析

对 7 个数据集进行分析，可知 NagoyaDataLeading, NagoyaOrigami 数据集前后帧之间变动较小，与视频实际的帧间变化类似；而其他数据集中变化较大，尤其是 Boys 数据集中人物和玩具前后变化幅度过大，不适合用前后帧关系去预测。

### 2.4 空域帧间分析

对 7 个数据集进行分析，可知 Boys, NagoyaDataLeading, NagoyaFujita, Toys 和 Trees 中的上下视角、左右视角变换比较均匀，可以认为镜头阵列与拍摄物体平面保持一致，因此可以做平移假设进行预测。

在 Cam\_Still 和 NagoyaOrigami 数据集中，明显发现镜头阵列所处的平面与拍摄物体平面不平行，物体离镜头越近在图片内的变动幅度越大，离镜头越远变动幅度越小，因此不适合做平移假设进行预测。

### 2.5 镜头分析

由于相机多视点成像特点，左右透镜所拍内容会与中间透镜的成像相似，但不完全相同，存在水平视差关系；上下透镜同理，如图 1.4 所示。

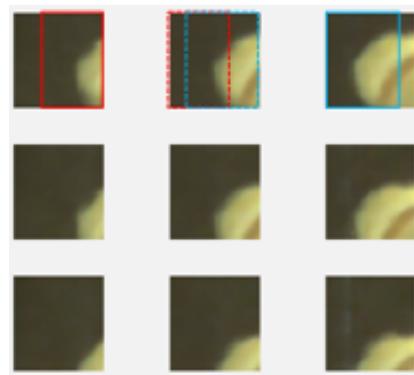


图 1.4 各方向视差图

### 3 模型设计概述

#### 3.1 创新点

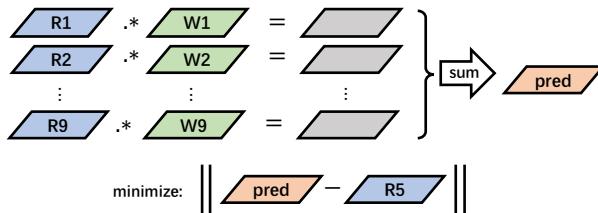
我们在这次项目上主要有如下的创新：提出了能很好解决帧间变化不大、场景复杂数据的 weight 模型；提出了能很好学到视差变化、在各个图片数据上都有很好表现的 transformer 模型；提出了能减小最优集数目大幅提升预测结果的微位移方法；提出了平移假设和 psnr 假设作为模型的理论基础，并同时减少计算量。

##### 3.1.1 weight 模型概述

本文的 weight 模型为周边 8 视角图片的每个像素的每个通道设一个权重变量，8 个视角图片乘以权重变量后相加得到一个预测的图片结果。权重的寻找过程是在上一帧的 9 个图片中进行，优化目标为 minimize 预测的图片与中间图片 P5 的距离。

在此之上，本文的 weight 模型加上了本文提出的微位移计算方法，减少了最优解，减少了权重变量求解的随机性，大幅提升了图片的预测结果。

weight 模型基本结构如下图所示：



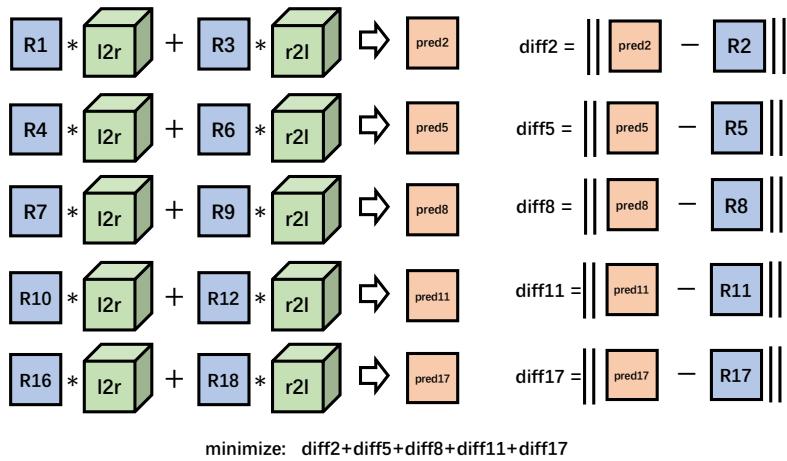
模型具体的细节见下一章节。

注：下一章节中的 4 代模型与 weight 模型对应，是 weight 模型与微位移方法的结合。

##### 3.1.2 transformer 模型概述

本文的 transformer 模型实现了用 cvx 求解视角的变化，用一个 3 维立方矩阵代表视角的变化，求得一个从左侧视角图往右侧变化的  $l2r\_transformer$  以及一个从右往左侧变化的  $r2l\_transformer$ 。优化的目标为 minimize 转化后的图片与中间目标图片的距离。求解得到 3 维变化矩阵后，将其应用于下一帧的第二行图像上，实现对中间图片的预测。

transformer 模型基本结构如下图所示：



模型具体细节见下一章节。

注：下一章节的 6 代模型与 transformer 模型对应，是 transformer 模型与微位移方法的结合。

### 3.1.3 微位移

本文提出的微位移方法通过对每个 block 多取周围一定的像素扩大原始 block，在凸优化模型求解时使用原始 block 的窗口在扩大后的 block 上位移滑窗，得到多个距离，将这些距离相加作为最后要优化缩小的目标，实现类似于机器学习中的扩展训练数据的过程。

微位移方法能大幅减小最优集数目、减小预测结果的不稳定性、使模型的结果能更加逼近最优解。本文的 weight 模型与 transformer 模型与微位移的方法融合，使得要求解的变换更加准确，生成的图像与目标图像更加接近。

- **背景：**

在 weight 模型中，每个位置的相同像素的相同通道的值、对应的变量和目标图片对应位置的值构成了一个超平面。因此写成优化目标如下：

$$\text{minimize: } \|a_1^T x_1 - b_1\| + \dots + \|a_n^T x_n - b_n\|$$

其中， $a$  为周围 8 视角图片的相同位置像素相同通道的值构成的向量， $x$  为对应的优化变量， $b$  为目标图片对应位置像素对应通道的值。

考虑单个像素单个通道情形，优化目标如下：

$$\text{minimize: } \|a_1^T x_1 - b_1\|$$

显然得到最优解时，目标函数的值为 0，此时最优解是一个超平面。

而不同像素不同通道的变量之间在模型中没有关系，因此原始优化问题（考虑所有的像素与通道）的目标函数的最优解也为 0，每个范数的值都为 0。此时  $x_1$  的解是一个超平面， $x_2$  的解是一个超平面… $x_n$  的解是一个超平面。

因此解的范围过大，使得模型随机性很大、不稳定，在预测中出现大量的预测错误部分。

- **假设：相似位置 block 对应的变化相同**

为了解决最优解的范围过大、模型随机性大、不稳定的问题，我们做出了如下的假设：

1. 得到一个 block
2. 将 block 往左或往右任意移动小范围的像素单位（1~4 个像素），得到新的 block
3. 这两个 block 上的求解的变化结果相似
4. 因此假设在新的 block 上要求解的凸优化模型与原始 block 上求解的优化变量结果相同。

- **方法细节：**

基于相似位置 block 对应的变化相同的假设，我们可以得到一个 block 周围的 n 个微位移后的 block，构成了 n 个样本，在这 n 个样本上实现相同的凸优化模型。因此此时凸优化问题可以描述如下：

$$\text{minimize: } \|a_{1,1}^T x_1 - b_{1,1}\| + \|a_{1,2}^T x_1 - b_{1,2}\| + \dots + \|a_{1,m}^T x_1 - b_{1,m}\|$$

$$+ \|a_{2,1}^T x_2 - b_{2,1}\| + \|a_{2,2}^T x_2 - b_{2,2}\| + \dots + \|a_{2,m}^T x_2 - b_{2,m}\|$$

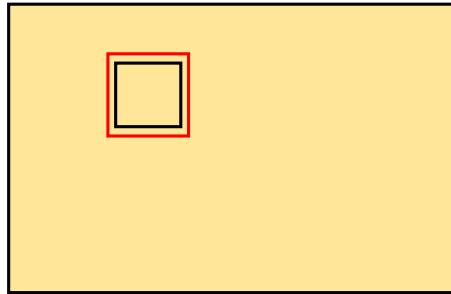
... ...

$$+ \|a_{n,1}^T x_n - b_{n,1}\| + \|a_{n,2}^T x_n - b_{n,2}\| + \cdots \|a_{n,m}^T x_n - b_{n,m}\|$$

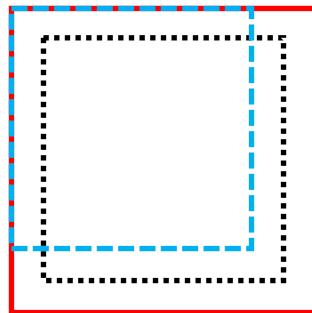
每一行由单个像素的单个通道构成，这样一个优化变量就由  $m$  个范数进行约束，减少了最优解的范围。

具体的实现过程如下：

1. 在原始的 block (内部黑色框) 上得到扩展的 block (红色方形)：



2. 在扩展 block 上按照原始 block 大小进行从左往右、从上往下的滑窗，得到  $m$  个 block 集：



其中红色方形为扩展 block，黑色为原始，蓝色虚线为最开始的 block，将蓝色虚线方形进行滑动得到  $m$  个 block 集。

3. 在得到的  $m$  个 block 集上进行求解凸优化问题，即

$$\text{minimize: } \|a_{1,1}^T x_1 - b_{1,1}\| + \|a_{1,2}^T x_1 - b_{1,2}\| + \cdots \|a_{1,m}^T x_1 - b_{1,m}\|$$

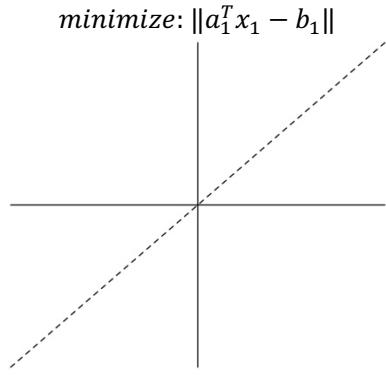
$$+ \|a_{2,1}^T x_2 - b_{2,1}\| + \|a_{2,2}^T x_2 - b_{2,2}\| + \cdots \|a_{2,m}^T x_2 - b_{2,m}\|$$

... ...

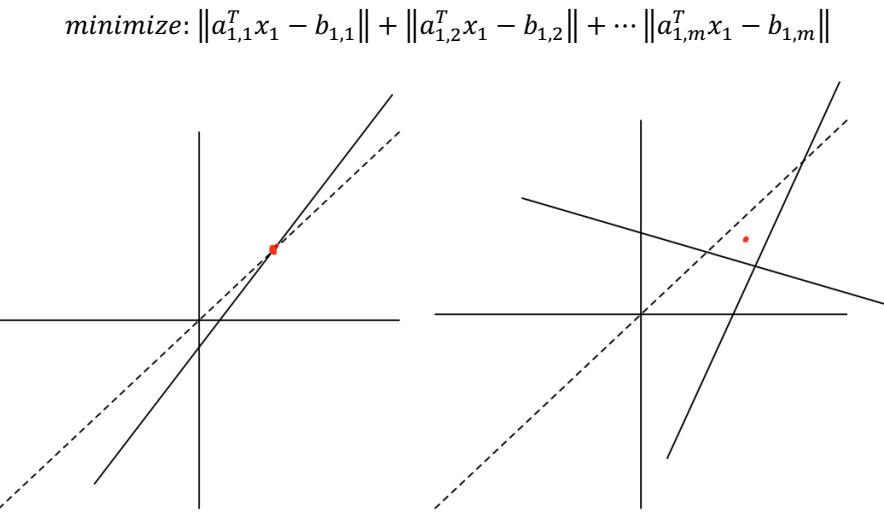
$$+ \|a_{n,1}^T x_n - b_{n,1}\| + \|a_{n,2}^T x_n - b_{n,2}\| + \cdots \|a_{n,m}^T x_n - b_{n,m}\|$$

### ● 数学解释：

以单个像素单个通道为例，其最优解几何解释是一条直线，如下图所示：



而使用微位移扩展后，每个范数构成一条直线，因此优化目标就是找出离这些直线最近的点。如下图两个例子所示：



上图为两条直线、三条直线的两种可能情形，其最优集都变成了一个点。直线平行时则最优集仍是平行线，但一般微位移后得到的直线的系数都有差别，很难构成平行的情况。因此扩展到高维空间中，最优集的范围可以被缩小。

### 3.1.4 使用凸优化模型求得视差的变换

本文的 weight 模型与 transformer 模型都是使用凸优化模型求得视差的变换，用不同的矩阵来实现了视差的变化，并未使用任何传统图像处理的滑窗方式搜索运动矢量来实现视差变换。

## 3.2 前提与假设

本文在实现模型时进行了 block 划分预处理，并假设周围 8 视角的 block 的 psnr 平均值与中间 block 的 psnr 值相同，以及假设行（列）方向的不同视角图相同行（列）处于相同的现实中的行（列），有相交部分，一个视角的某一行可通过在该上平移得到与另一视角同一行的部分内容。

### 3.2.1 block 划分

本文将每张图片划分成小的 block，凸优化模型在每个小 block 上进行求解。本文使用的 block 大小有  $8 \times 8$  与  $16 \times 16$  两种。

划分成 block 有如下好处：

1. 减少计算时间：

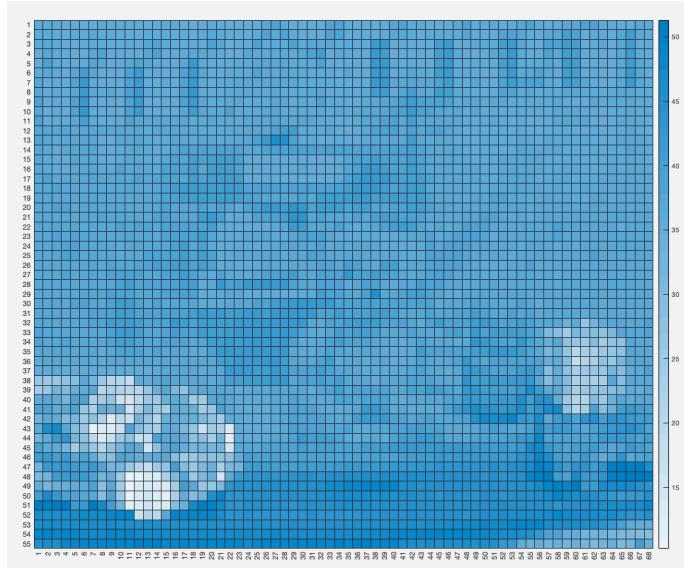
block 越大，使得凸优化模型的参数数量增长越快，导致单个像素计算时间呈指数上升趋势，因此划分成 block 能减少计算时间。

2. 应用 psnr 假设：

block 划分后，可以结合下一点的 psnr 假设，使得 psnr\_mat 上 block 对应的 psnr 较高的块直接使用上一帧的图像，不进行凸优化求解，减少计算时间。

### 3.2.2 psnr 假设

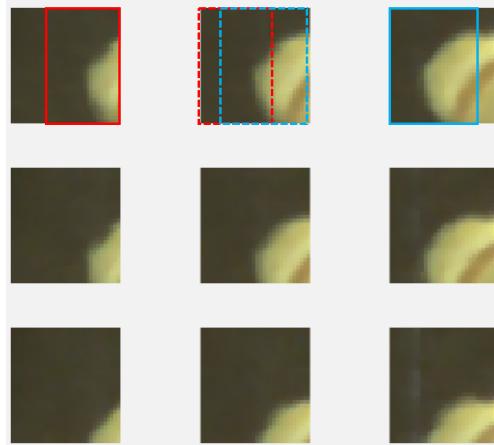
本文假设在划分成 block 后，周围 8 个视角的 block 前后帧的 psnr 与中间视角的 psnr 值保持一致，因此计算周围 8 个视角每个 block 上的 8 视角平均 psnr 当作中间视角前后帧的 psnr 值，psnr 图可视化举例如下：



白色部分是前后两帧之间 psnr 值较高的 block，并且与实际的中间视角图片的前后两帧变化的部分相对应，证明了 psnr 假设成立。

### 3.2.3 平移假设

考虑到相机镜头的机位排列，本文假设行（列）方向的不同视角图相同行（列）处于相同的现实中的行（列），有相交部分，一个视角的某一行可通过在该上平移得到与另一视角同一行的部分内容。



以上图为例，图中的每一行的三个图片中，可以简单地将左边和右边的图片左右平移和裁切得到相似的中间图片（左上角裁切出红色部分平移到中间的红色虚线处，右上角蓝色部分同样处理到中间蓝色虚线处，拼凑成预测的中间图片），每一列类似的能从上下得到中间的图片。

### 3.3 设计过程

我们设计的思路与过程主要经历了尝试求坐标变换、矩阵间接实现平移变换、weight 模型权重实现变换、transformer+权重实现时域变换、transformer 实现空域视角变换得到最终的几个模型。

#### 1. 求坐标变换矢量：

我们最早想通过光学几何中的仿射、投影、射影等变换来实现周围视角的图片变换到中间视角上。但是这种方式的优化变量是对应的几何变换矩阵，是在像素点的坐标上进行，使得凸优化模型的矩阵索引出现优化变量，导致 cvx 求解器无法求解，因此放弃了该方法。

#### 2. 简单权重：

为了有一个 baseline 模型，我们在最开始设计了一个简单权重模型，周围视角每张图都有一个权重变量，8 个视角总计 8 个权重，每个视角图片与权重相乘后相加得到预测图片，最小化预测图片与目标图片的距离。

#### 3. 矩阵间接实现平移变换：

随后我们进行平移假设，最开始的实现是一行的右侧视角预测中间图片的右半部分，左侧视角图预测中间图片的左半部分，左右分离，导致图片的结果很差，出现光栅状的情形。

#### 4. weight 权重实现变换：

由于前面的方法效果欠佳，我们尝试用权重的形式实现预测。对每个像素每个通道赋予一个权重变量，优化该权重变量。在这个过程中，为了解决结果不稳定性问题，我们还提出了微位移的方法减小最优集。

#### 5. weight 权重+时域变换：

虽然 weight 权重的方法在前后帧变动范围较小的时候预测的表现优异，但是在运动幅度大且不规则的时候，会出现预测失常，出现上一帧的信息。因此我们考虑添加时域变换的矩阵，对上一帧的图片进行变换后作为近似的目标来实现时域上的变化。但由于近似假设与实际情况有所偏离，因此大幅度变动场景预测结果依然不佳。

#### 6. transformer 空域变换：

为了解决变动较大预测不精准的问题，我们回到了最开始的平移假设，使用 3 维立方矩阵来做平移上的变换，并且将上下、左右的图片进行联合优化，而非最开始的左右单独分开，使得模型最终在各个场景都有很好的预测结果。

## 4 模型细节

我们在整个实验流程中，从权重到视差空域变换总共设计了 6 个模型，其中第 4 个 weight 模型与第 6 个 transformer 模型有较好的的结果。下文阐述每个模型的技术细节。

### 4.1 简单权重模型

- 设计目的：

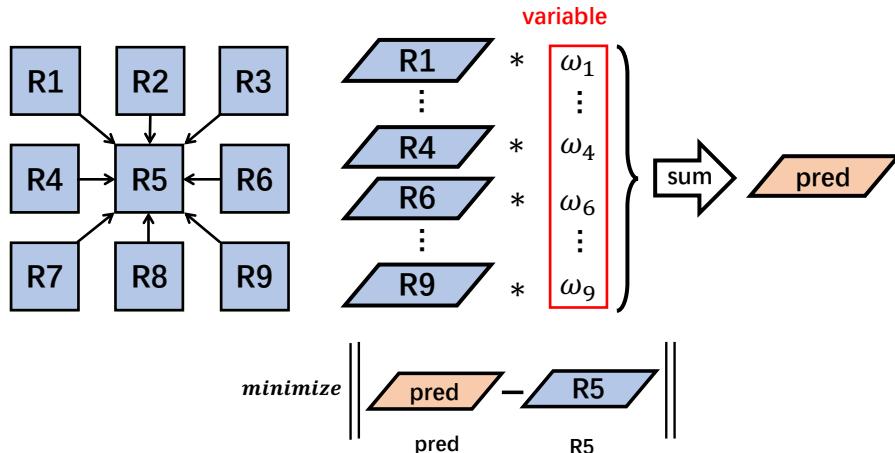
我们在实验最开始设计了简单权重模型来初步验证权重的可行性，并且作为 baseline 与后面 weight 权重模型进行对比。

- 模型结构：

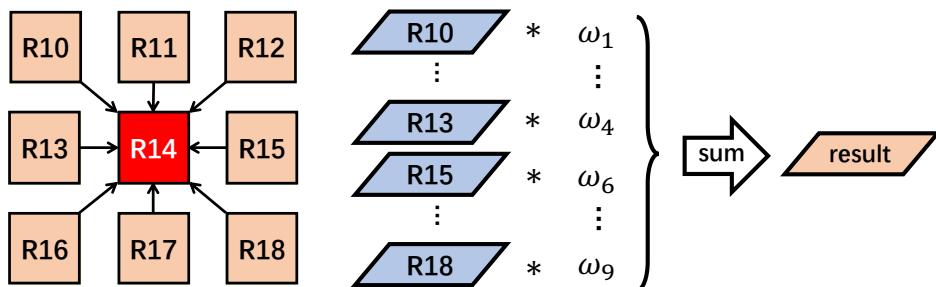
简单权重模型对周围视角每张图都设置一个权重变量，8 个视角总计 8 个权重，每个视角图片与权重相乘后相加得到预测图片，最小化预测图片与目标图片的距离，以得到权重变量。模型的结构分为从上一帧求解 weight 权重、用求解得到的权重计算出下一帧的目标 X 两个部分。此处，简单权重模型设计为整体加权和分块加权两种方式，两者原理相同，区别在于是否进行分块处理。

- (1) 整体加权模型

从上一帧求解凸优化问题得到权重的过程如下所示：



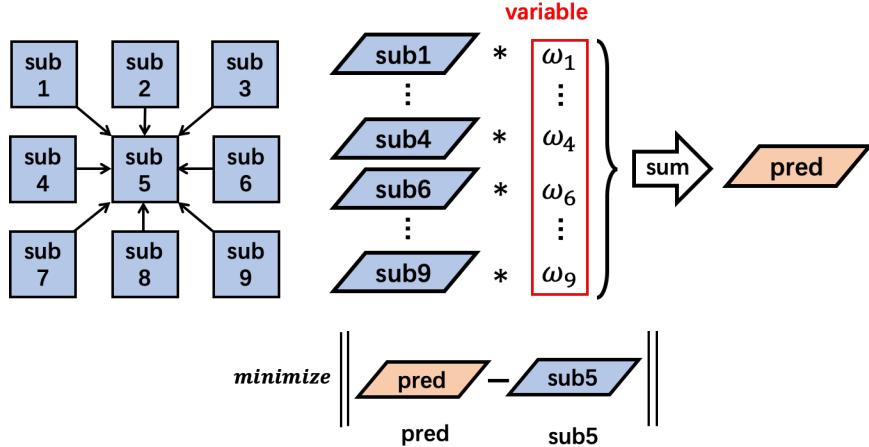
求解过程时， $\omega$  为优化变量，而在计算下一帧目标  $R_{14}$  时，所有的量都为确定量。根据权重计算出下一帧的目标  $R_{14}$  过程与求解过程类似，如下所示：



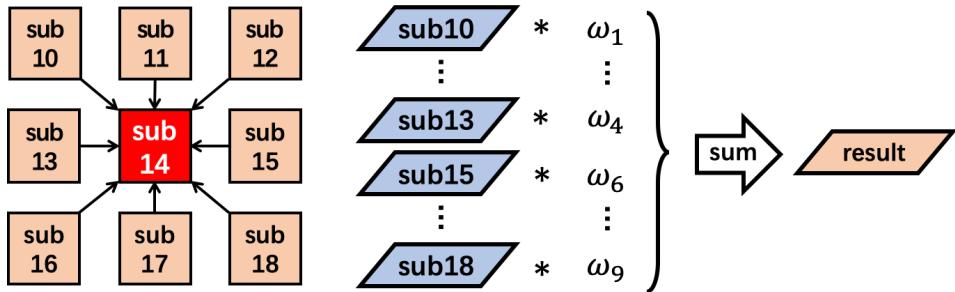
通过两个过程得到预测的目标。

## (2) 分块加权模型

此模型与上面(1)中模型类似，从上一帧求解凸优化问题得到权重的过程如下所示：



求解过程时， $\omega$ 为优化变量，而在计算下一帧目标  $R14$  时，所有的量都为确定量。根据权重计算出下一帧的目标  $R14$  过程与求解过程类似，如下所示：



同理，通过这两个过程得到预测的目标。

### ● 公式化描述：

#### (1) 整体加权凸优化模型

整体加权模型的数学描述为， $R1, \dots, R13, R15, \dots, R18$  分别为参考图像的三维矩阵，令  $\text{train} = [R1, \dots, R4, R6, \dots, R9]$ ,  $\text{test} = [R10, \dots, R13, R15, \dots, R18]$ ，设  $\text{weight} \in R^{8 \times 1}$

$$\begin{aligned} & \text{minimize } \parallel \text{train} * \text{weight} - R5 \parallel \\ & \text{subject to} \\ & 0 \leq \text{weight} \leq 1 \\ & \text{sum}(\text{weight}) == 1 \end{aligned}$$

此时预测图像即为， $\text{pred}_{img} = \text{test} * \text{weight}$ 。

#### (2) 分块加权凸优化模型

分块加权模型是将参考图像划分成小块 block 进行训练和预测，同理， $\text{sub\_img1}, \dots, \text{sub\_img13}, \text{sub\_img15}, \dots, \text{sub\_img18}$  分别为参考图像中相应 block 的三维矩阵，令  $\text{train} = [\text{sub\_img1}, \dots, \text{sub\_img4}, \text{sub\_img6}, \dots, \text{sub\_img9}]$ ,  $\text{test} = [\text{sub\_img10}, \dots, \text{sub\_img13}, \text{sub\_img15}, \dots, \text{sub\_img18}]$ ，设  $\text{weight} \in R^{8 \times 1}$

$$\begin{aligned} & \text{minimize } \parallel \text{train} * \text{weight} - \text{sub}_{img5} \parallel \\ & \text{subject to} \end{aligned}$$

$$0 \leq weight \leq 1$$

$$\sum(weight) == 1$$

此时预测 block 图像即为,  $pred_{sub\_img} = test * weight$ , 将所有预测的 block 拼接起来就是预测的整个图像。

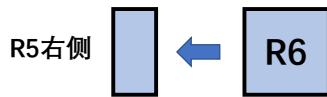
注意：之后所示方法均为分块实现，后面不做赘述。

## 4.2 矩阵平移变换模型 v1

- **设计目的：**

根据前面的平移假设，我们希望通过设计一个变换矩阵，让其与原图在运算后实现平移的变换。考虑到左右、上下视角的图片有相似的对称性，我们将其看作相同的变换方式，通过旋转、翻转等操作能相互等价。

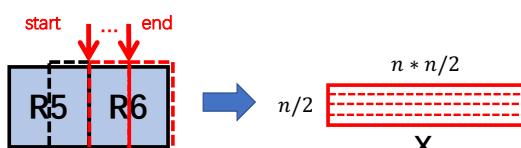
以右侧视角图为例，我们假设通过相应变换能得到中间视角图的右半部分，如下图所示：



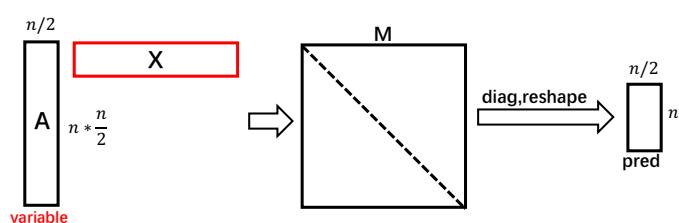
因为我们假设上下左右变换方式相同，因此将左、上、下侧视角图变换到与右侧视角图一致的位置，与右侧视角图使用相同的变换矩阵。

- **模型结构：**

以 R5 和 R6 之间的变换为例，假设 block 的边长为  $n$ ，从 R6 的左半部分开始往右按像素平移，每次将对应的  $n * \frac{n}{2}$  的矩阵展成向量，平移  $\frac{n}{2}$  次总共得到  $\frac{n}{2}$  个向量，将这些向量组合成  $[\frac{n}{2}, \frac{n}{2} * n]$  大小的矩阵 X，如下图所示：



构造  $[\frac{n}{2} * n, \frac{n}{2}]$  的变换矩阵 A。矩阵 A 与矩阵 X 相乘后得到方阵 M，取方阵 M 的对角线元素作为预测的 R5 的右半部分，如下图所示：



对变换后的上、下、左侧视角使用相同的矩阵 A 做同样的变换，因此整个凸优化模型如下所示：

$$\text{minimize} \quad \left\| \begin{array}{c} \text{pred右} \\ - \\ \text{R5右} \end{array} \right\| + \left\| \begin{array}{c} \text{pred左} \\ - \\ \text{R5左} \end{array} \right\| + \left\| \begin{array}{c} \text{pred上} \\ - \\ \text{R5上} \end{array} \right\| + \left\| \begin{array}{c} \text{pred下} \\ - \\ \text{R5下} \end{array} \right\|$$

使用 CVX 求解得到变换矩阵 A 后，将其应用于下一帧，使用同样的计算方式直接求得预测目标 R14。

- 公式化描述：

以图像单通道为例，right5 表示 R5 的右半侧图像，left5 表示 R5 左半侧图像，up5 表示 R5 上半侧图像，down5 表示 R5 下半侧图像。 $X_6$ ,  $X_4$ ,  $X_2$  和  $X_8$  分别表示以上述逐像素划窗方式所得  $R6$ ,  $R4$ ,  $R2$  和  $R8$  的图像内容，它们的大小均为  $(n/2, n \times n/2)$ ，设变量  $A_6$ ,  $A_4$ ,  $A_2$  和  $A_8$  为变换矩阵，大小均为  $(n \times n/2, n/2)$ 。

$$\begin{aligned} X'_6 &= \text{diag}(A_6 * X_6), \text{pred}_6 = \text{reshape}(X'_6) \\ X'_4 &= \text{diag}(A_4 * X_4), \text{pred}_4 = \text{reshape}(X'_4) \\ X'_2 &= \text{diag}(A_2 * X_2), \text{pred}_2 = \text{reshape}(X'_2) \\ X'_8 &= \text{diag}(A_8 * X_8), \text{pred}_8 = \text{reshape}(X'_8) \end{aligned}$$

$$\begin{aligned} \text{minimize} \quad & ||\text{pred}_6 - \text{right5}|| + ||\text{pred}_4 - \text{left5}|| \\ & + ||\text{pred}_2 - \text{up5}|| + ||\text{pred}_8 - \text{down5}|| \\ \text{subject to} \quad & \end{aligned}$$

$$\begin{aligned} 0 \leq A_2 &\leq 1 \\ 0 \leq A_4 &\leq 1 \\ 0 \leq A_6 &\leq 1 \\ 0 \leq A_8 &\leq 1 \end{aligned}$$

最后通过下一帧 R11, R13, R15 和 R17 四张图像信息来预测 R14。

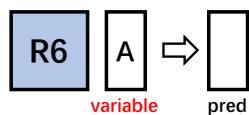
### 4.3 矩阵平移变换模型 v2

- 设计目的：

前面的矩阵平移变换模型 v1 由于在构造  $X$  矩阵时是采用滑窗方式，每滑动一次取出窗内所有元素构成矩阵的一行，这导致了当 block 的大小增长时，消耗的内存与创建的变换矩阵 A 呈幂次增长，需要大量的计算资源。为了解决这一问题，我们对原先的平移变换模型进行了改进，得到了矩阵平移变换模型 v2。

- 模型结构：

矩阵变换模型 v2 与 v1 的区别在于， $X$  直接以  $R6$  进行代替，而 A 矩阵则是  $[n, \frac{n}{2}]$  大小的矩阵。 $XA$  后得到  $[n, \frac{n}{2}]$  大小的预测的 R5 的右半部分，相应的结构图如下所示：



同时将 A 应用于上下左右，得到最终的优化目标：

$$\text{minimize} \quad \left\| \begin{array}{c} \text{pred右} \\ - \\ \text{R5右} \end{array} \right\| + \left\| \begin{array}{c} \text{pred左} \\ - \\ \text{R5左} \end{array} \right\| + \left\| \begin{array}{c} \text{pred上} \\ - \\ \text{R5上} \end{array} \right\| + \left\| \begin{array}{c} \text{pred下} \\ - \\ \text{R5下} \end{array} \right\|$$

使用 CVX 求解得到变换矩阵 A 后，将其应用于下一帧，使用同样的计算方式直接求得预测目标 R14。

- 公式化描述：

仍以单通道为例，right5 表示 R5 的右半侧图像，left5 表示 R5 左半侧图像，up5 表示 R5 上半侧图像，down5 表示 R5 下半侧图像。设变量  $A_6, A_4, A_2$  和  $A_8$  为变换矩阵，大小均为  $(n, n/2)$ 。

$$\text{minimize} \quad ||R6 * A_6 - right5|| + ||R4 * A_4 - left5||$$

$$+ ||R2 * A_2 - up5|| + ||R8 * A_8 - down5||$$

*subject to*

$$0 \leq A_2 \leq 1$$

$$0 \leq A_4 \leq 1$$

$$0 \leq A_6 \leq 1$$

$$0 \leq A_8 \leq 1$$

最后通过下一帧 R11, R13, R15 和 R17 四张图像信息来预测 R14。

## 4.4 weight 模型（较优）

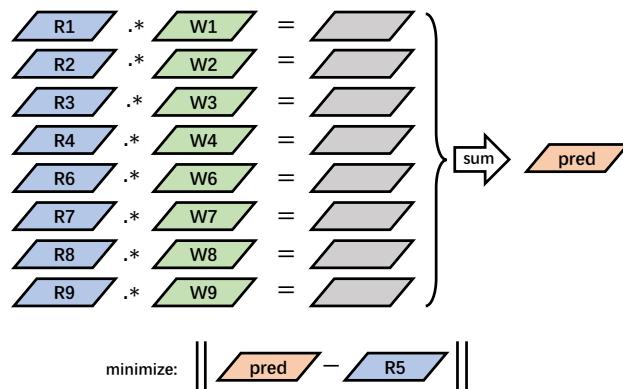
- 设计目的：

平移变换模型 v1 与 v2 在预测的结果上都有明显的光栅状痕迹。考虑到最开始简单权重模型能有较为均匀的结果，我们转换思路，在简单权重模型上进行扩展，对所有的像素的每个通道设定相应的权重，并通过微位移方法减小最优集的范围，减少结果的不确定性。

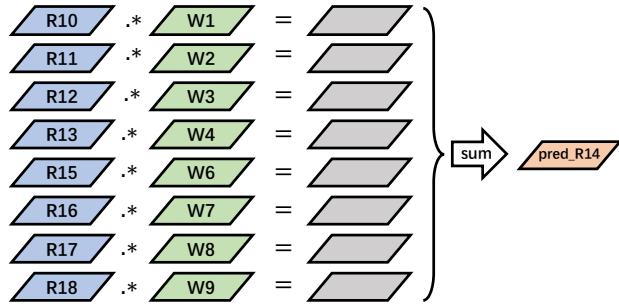
- 模型结构：

weight 模型对周围视角的每个图片的所有像素的所有通道都创建相应的权重变量，与变量相乘后将结果相加得到预测的结果。为了方便描述，以单通道为例，模型的结构分为从上一帧求解 weight 权重、用求解得到的权重计算出下一帧的目标 X 两个部分。

从上一帧求解 weight 权重如下所示：



求解过程时,  $W_n$  为优化变量, 而在计算下一帧目标  $R_{14}$  时, 所有的量都为确定量。根据权重计算出下一帧的目标  $R_{14}$  过程与求解过程类似, 如下所示:



通过两个过程得到预测的目标。

注: weight 模型还结合了微位移, 为了方便说明没有添加微位移在模型图中。

- 公式化描述:

结合微位移方法, 假设共有  $m$  个滑窗, 则对应有  $m$  组样本集,  $R_1, \dots, R_{13}, R_{15}, \dots, R_{18}$  分别为参考图像 block 的三维矩阵, 设  $W_1, \dots, W_4, W_6, \dots, W_9$  为权重矩阵。

$$\begin{aligned} & \text{minimize } ||R_{11}.* W_1 + \dots + R_{18}.* W_9 - R_{51}|| + \\ & \quad \dots \\ & \quad + ||R_{1m}.* W_1 + \dots + R_{9m}.* W_9 - R_{5m}|| \\ & \text{subject to} \end{aligned}$$

$$0 \leq W_i \leq 1, i = 1, \dots, 4, 6, \dots, 9$$

此时预测图像即为,  $\text{pred}_{img} = R_{10}.* W_1 + \dots + R_{18}.* W_9$

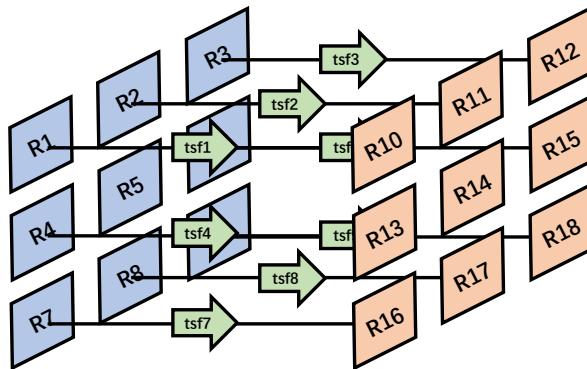
## 4.5 帧间时域变化模型 (试验)

- 设计目的:

虽然 weight 权重的方法在前后帧变动范围较小的时候预测的表现优异, 但是在运动幅度大且不规则的时候, 会出现预测失常, 出现上一帧的信息。因此我们考虑添加时域变换的矩阵, 对上一帧的图片进行变换后作为近似的目标准来实现时域上的变化, 尝试通过时域上的变换学习将多出的上一帧信息消除。

- 模型结构:

我们通过  $R_1$  预测  $R_{10}$ 、 $R_2$  预测  $R_{11}$ …… $R_9$  预测  $R_{18}$  来求解时域得到的变化信息, 如下图所示, 每个变换有一个对应矩阵:



模型求解结构分为 4 部分，第 1 部分求解变换矩阵，第 2 部分根据求解出的变换矩阵构造变换后的预测目标，第 3 部分应用 weight 模型使前一帧的周边视角预测结果逼近第 2 步的构造出的预测目标，第 4 部分用第 3 部分中的变换矩阵计算得到最终目标。每个部分具体的结构图如下所示：

第 1 部分：

我们通过下图所示的变换，对周围的每一个视角图求解从上一帧变换到下一帧的变换矩阵。

$$\begin{array}{l}
 \begin{array}{ccc}
 R1 & \cdot\cdot\cdot & tsf1 \\
 R2 & \cdot\cdot\cdot & tsf2 \\
 R3 & \cdot\cdot\cdot & tsf3 \\
 R4 & \cdot\cdot\cdot & tsf4 \\
 R6 & \cdot\cdot\cdot & tsf6 \\
 R7 & \cdot\cdot\cdot & tsf7 \\
 R8 & \cdot\cdot\cdot & tsf8 \\
 R9 & \cdot\cdot\cdot & tsf9
 \end{array} = \begin{array}{c}
 pred10 \\
 pred11 \\
 pred12 \\
 pred13 \\
 pred15 \\
 pred16 \\
 pred17 \\
 pred18
 \end{array} \\
 \text{minimize: } \| \begin{array}{c} pred10 \\ - R10 \end{array} \| + \cdots + \| \begin{array}{c} pred18 \\ - R18 \end{array} \|
 \end{array}$$

第 2 部分：

在得到变换矩阵后，将其应用于周围的视角得到相应的变换结果集，然后对变换结果集求均值，得到 target。

$$\begin{array}{l}
 \begin{array}{ccc}
 R1 & \cdot\cdot\cdot & tsf1 \\
 R2 & \cdot\cdot\cdot & tsf2 \\
 \vdots & \vdots & \vdots \\
 R9 & \cdot\cdot\cdot & tsf9
 \end{array} = \begin{array}{c}
 pred10 \\
 pred11 \\
 \vdots \\
 pred18
 \end{array} \} \xrightarrow{\text{mean}} \text{target}
 \end{array}$$

第 3 部分：

与 weight 模型类似，将预测的目标换成第 2 部分中得到的 target。

$$\begin{array}{l}
 \begin{array}{ccc}
 R1 & \cdot\cdot\cdot & W1 \\
 R2 & \cdot\cdot\cdot & W2 \\
 \vdots & \vdots & \vdots \\
 R9 & \cdot\cdot\cdot & W9
 \end{array} = \begin{array}{c} \text{pred} \end{array} \} \xrightarrow{\text{sum}} \text{pred} \\
 \text{minimize: } \| \begin{array}{c} \text{pred} \\ - \text{target} \end{array} \|
 \end{array}$$

第 4 部分：

$$\begin{array}{l}
 \begin{array}{ccc}
 R10 & \cdot\cdot\cdot & W1 \\
 R11 & \cdot\cdot\cdot & W2 \\
 \vdots & \vdots & \vdots \\
 R18 & \cdot\cdot\cdot & W9
 \end{array} = \begin{array}{c} \text{pred} \end{array} \} \xrightarrow{\text{sum}} \text{pred14}
 \end{array}$$

注：weight 模型还结合了微位移，为了方便说明没有添加微位移在模型图中。

- 公式化描述：

用  $R1, \dots, R13, R15, \dots, R18$  分别为参考图像的三维矩阵，设  $tsf1, \dots, tsf4, tsf6, \dots, tsf9$  为对应前后帧之间的变换矩阵。

$$\begin{aligned} & \text{minimize } ||R1.* tsf1 - R10|| + \dots + ||R9.* tsf9 - R18|| \\ & \text{subject to} \end{aligned}$$

$$0 \leq tsfi \leq 1, i = 1, \dots, 4, 6, \dots, 9$$

分别求出  $tsf1, \dots, tsf4, tsf6, \dots, tsf9$ ，那么  $target = (R1.* tsf1 + \dots + R9.* tsf9)/8$ 。再结合微位移方法，假设共有  $m$  个滑窗，则对应有  $m$  组样本集， $R1, \dots, R13, R15, \dots, R18$  分别为参考图像 block 的三维矩阵，设  $W1, \dots, W4, W6, \dots, W9$  为权重矩阵。

$$\begin{aligned} & \text{minimize } ||R1_1.* W1 + \dots + R9_1.* W9 - target_1|| + \\ & \quad \dots \\ & \quad + ||R1_m.* W1 + \dots + R9_m.* W9 - target_m|| \\ & \text{subject to} \end{aligned}$$

$$0 \leq Wi \leq 1, i = 1, \dots, 4, 6, \dots, 9$$

此时预测图像即为  $pred_{img} = R10.* W1 + \dots + R18.* W9$ 。

## 4.6 transformer 模型（最优）

- 设计目的：

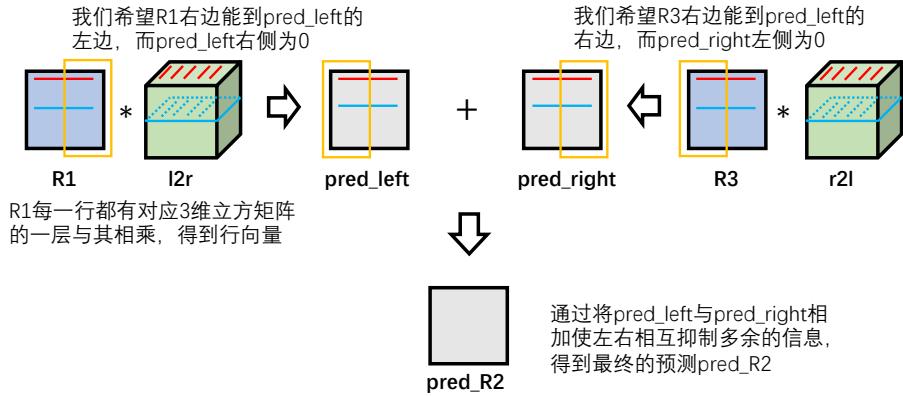
结合了微位移的 weight 模型虽然在大部分场景能有很高的结果，但在变动幅度特别大的时候会出现预测错乱的情况。为了解决这一问题，我们重新从视角变换这一出发点着手，结合平移假设、微位移和 weight 模型的思想，构造了 transformer 模型，通过视角变换立方阵，采用对称联合的方式解决了之前平移假设中左右分离时出现的光栅现象，在所有类型的图片中均有优异的预测性能。

- 模型结构：

transformer 模型有 3 种类型，分为水平视角的变换，垂直视角的变换、以及水平和垂直视角变换联合的模型。水平视角变换与垂直视角变换模型仅有方向上的差别，因此可看作同一模型，而水平与垂直视角联合变换的模型虽然预测结果很高，但比起单独的水平或垂直模型，消耗大量的时间。因此我们的 transformer 模型采用的是水平视角的变换。

水平视角变换的过程中，以同一行的  $R1, R2, R3$  为例，我们构造了一个  $l2r$  立方矩阵和  $r2l$  立方矩阵，分别实现  $R1$  预测  $R2$  的左侧部分， $R3$  预测  $R2$  的右侧部分。将两个预测结果  $pred\_left$  和  $pred\_right$  相加之后得到最终预测的  $pred\_R2$ ，如下图所示：

左



在  $R_1$  与立方矩阵  $l2r$ 、 $R_3$  与  $r2l$  的计算过程中,  $R_1$  的每一行都有对应的 3 微立方矩阵中的一层矩阵与其相乘得到一行向量, 将每一次相乘得到的向量堆叠得到  $\text{pred\_left}$ 、 $\text{pred\_right}$  矩阵。

将变换立方矩阵  $l2r$  和  $r2l$  同样按照行应用于  $\langle R_4, R_5, R_6 \rangle$ 、 $\langle R_7, R_8, R_9 \rangle$ 、 $\langle R_{10}, R_{11}, R_{12} \rangle$ 、 $\langle R_{16}, R_{17}, R_{18} \rangle$  上, 预测结果与目标相减求 2 范数, 得到  $\text{diff}_2$ ,  $\text{diff}_5$ ,  $\text{diff}_8$ ,  $\text{diff}_{11}$ ,  $\text{diff}_{17}$ , 将 5 个  $\text{diff}$  项相加作为优化目标, 通过凸优化求解器 cvx 求解得到  $r2l$  和  $l2r$  立方矩阵。

$R_1 * l2r + R_3 * r2l \Rightarrow \text{pred}_2$	$\text{diff}_2 = \ \text{pred}_2 - R_2\ $
$R_4 * l2r + R_6 * r2l \Rightarrow \text{pred}_5$	$\text{diff}_5 = \ \text{pred}_5 - R_5\ $
$R_7 * l2r + R_9 * r2l \Rightarrow \text{pred}_8$	$\text{diff}_8 = \ \text{pred}_8 - R_8\ $
$R_{10} * l2r + R_{12} * r2l \Rightarrow \text{pred}_{11}$	$\text{diff}_{11} = \ \text{pred}_{11} - R_{11}\ $
$R_{16} * l2r + R_{18} * r2l \Rightarrow \text{pred}_{17}$	$\text{diff}_{17} = \ \text{pred}_{17} - R_{17}\ $
$\text{minimize: } \text{diff}_2 + \text{diff}_5 + \text{diff}_8 + \text{diff}_{11} + \text{diff}_{17}$	

得到  $l2r$  和  $r2l$  立方阵后, 将其与  $R_{13}$  和  $R_{15}$  进行与求解过程类似的操作, 得到最终预测的目标  $R_{14}$ , 如下图所示:

$$R_{13} * l2r + R_{15} * r2l \Rightarrow \text{pred}_{14}$$

### ● 公式化描述:

仍以单通道为例,  $R_1, \dots, R_{12}, R_{16}, \dots, R_{18}$  为参考图像, 设  $l2r$  和  $r2l$  分别为从左往右的三维变换矩阵和从右往左的三维变换矩阵。

$$\text{minimize } \|R_1 l2r + R_3 r2l - R_2\| + \dots + \|R_{16} l2r + R_{18} r2l - R_{17}\|$$

*subject to*

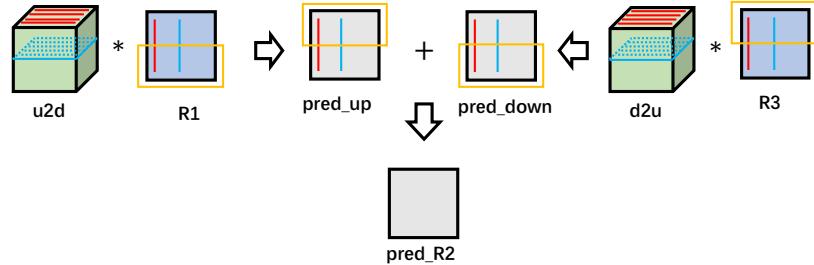
$$0 \leq l2r \leq 1$$

$$0 \leq r2l \leq 1$$

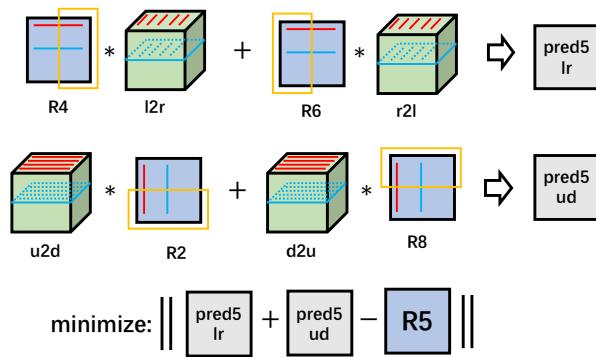
求得最优变换矩阵  $l2r$  和  $r2l$ , 再带入求得  $\text{pred} = R13 l2r + R15 r2l$ 。

- 垂直与水平变换联合优化：

垂直变换的模型计算方式与水平变换的计算方式类似，如下图所示：



将垂直和水平在上一帧的中间视角图  $R5$  进行联合：



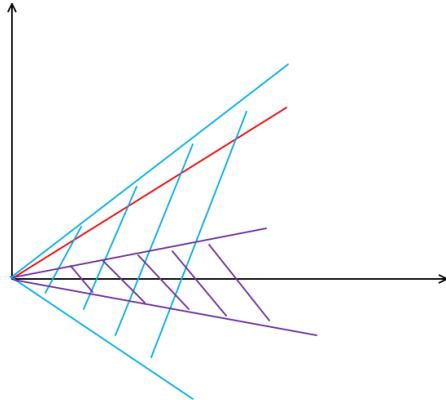
上图只展示了垂直与水平之间的联合变换部分，模型中的其他正则项还包括独立的水平变换与垂直变换。

- 对称相加的目的：

在前面的结构图中可以看到不论是水平还是垂直方向的模型，我们都采取两侧预测的结果相加后再减去预测目标，最后求范数，而不是两侧分别单独与结果相减、分别求范数再将范数相加。这是因为下面几个原因之一：

1. 我们的平移变换模型采用了左右独立的思想，导致出现了严重的光栅现象，说明不能将左右变换分离，也不能将左右的变换看作独立。
2. 每一侧的图片只有中间图片的部分信息，两侧相加后才包括中间图片所需要的所有信息。
3. 以水平变换为例，左右的变换可以相互监督，右侧变化能抑制左侧变换结果右侧的噪声部分，而左侧变化能抑制住右侧变换结果左侧的噪声部分。

相互监督的数学解释如下：



在左往右变换结果的右半部分，因为左侧视角图并不包含中间视角右半部分信息，我们希望其结果为0，让右往左变换的结果去贴近真实目标。图中红色的超平面是我们希望获得的目标，蓝色的是右往左可能的变换结果，紫色的是左往右可能的结果。因为右往左包含中间视角右侧的信息，而左往右不包含，因此可以定性的认为右往左更容易拟合红色的目标，在图中我们将其解释为蓝色部分包含红色部分，而左往右的紫色部分则不包含红色目标。

当单独只有紫色部分时，模型 minimize 出的最好结果左往右为紫色的上部边，并非为0，而右往左则是拟合了红色的直线，但此时二者 minimize 的结果相加会导致偏离红色直线。因此通过将左右联合在一起 minimize，更容易使得使紫色部分为0、蓝色部分拟合红色部分，使二者结果相加更容易拟合红色的目标直线。

## 5 模型结果

### 5.1 transformer 模型

transformer 模型在 7 个数据集上均有突出的结果表现。其中 2 个数据集的 PSNR 值在 52 以上，3 个数据集的 PSNR 值在 40 以上，而 Toys 的结果为 37.6 逼近 40。另外 Trees 的结果为 28.28，主要是由于场景变动幅度大、图片内容纹理复杂导致 PSNR 稍低，但是 Trees 图片预测结果肉眼难以与真实结果区分，结果较好。

- PSNR 结果：

PSNR 结果如下：

表 5.1 transformer 模型 PSNR 结果表

model	Boys	Cam_still	Leading	Fujita	Origami	Toys	Trees	Average
transformer <sup>1</sup>	42.2212	41.0064	52.6149	52.7888	49.8917	35.2635	28.2809	43.1525
transformer <sup>2</sup>	42.2212	41.0174	52.6149	52.7888	49.8917	37.5560	28.2809	43.4816
baseline	22.0778	24.2851	45.9011	38.2991	43.2356	30.8380	12.2819	30.9884

注：transformer<sup>1</sup> 中计算并优化全部的块；transformer<sup>2</sup> 中每张图选取最优的 thresh，对 psnr\_mat 中低于设定的 thresh 进行优化，计算优化全部的块能往往能获得更好的结果，但时间消耗更大。

baseline 为直接拿上一帧 R5 作为结果与 R14 计算 PSNR 值；psnr\_mat 请见前文 psnr 假设，最优 PSNR 的选择见 thresh 对 transformer 结果影响的对比实验部分。

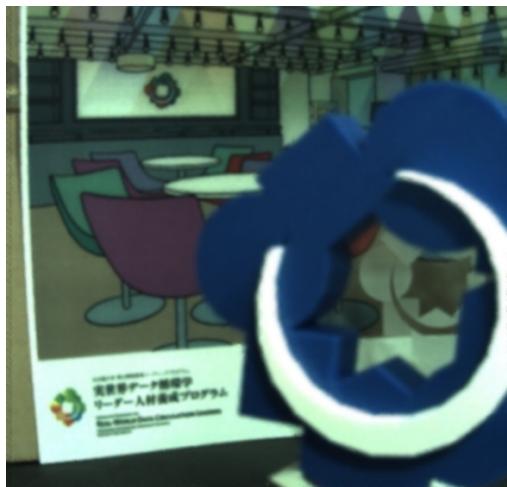
表 5.1 中，transformer<sup>1</sup> 与 transformer<sup>2</sup> 的 block 大小均为 16，微位移 pad\_size 量为 1。对所有块进行计算优化的 transformer<sup>2</sup> 比 transformer<sup>1</sup> 有着更为突出的性能，反映了模型的最优性能。而 transformer<sup>1</sup> 的结果则反映了考虑时间消耗时的模型最优性能。

- 图片结果：

transformer 模型在所有的数据上的预测图片结果展示，大图请见一同上传的图片文件。

表 5.2 transformer 模型图片结果

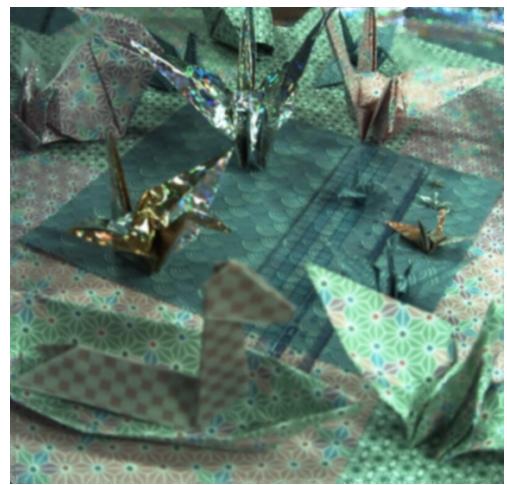
Boys	Cam_Still
	
NagoyaDataLeading	NayoyaFujita



NagoyaOrigami



Toys



Trees



-



## 5.2 weight 模型

weight 模型除了在 Boys 集上没法做的较好的预测外，其他图片集上都有较好的结果，不论是 PSNR 指标还是实际肉眼观察结果都有良好的表现。其中 Trees 的结果为 28.8066，主要是由于场景变动幅度大、图片内容纹理复杂导致 PSNR 稍低，但是 Trees 图片预测结果肉眼难以与真实结果区分，结果较好。

- PSNR:

psnr 结果如下

表 5.3 weight 模型 PSNR 结果表

model	Boys	Cam_still	Leading	Fujita	Origami	Toys	Trees	Average
weight	29.1528	38.3948	46.1306	44.6220	44.0467	37.9978	28.8066	38.4502
baseline	22.0778	24.2851	45.9011	38.2991	43.2356	30.8380	12.2819	30.9884

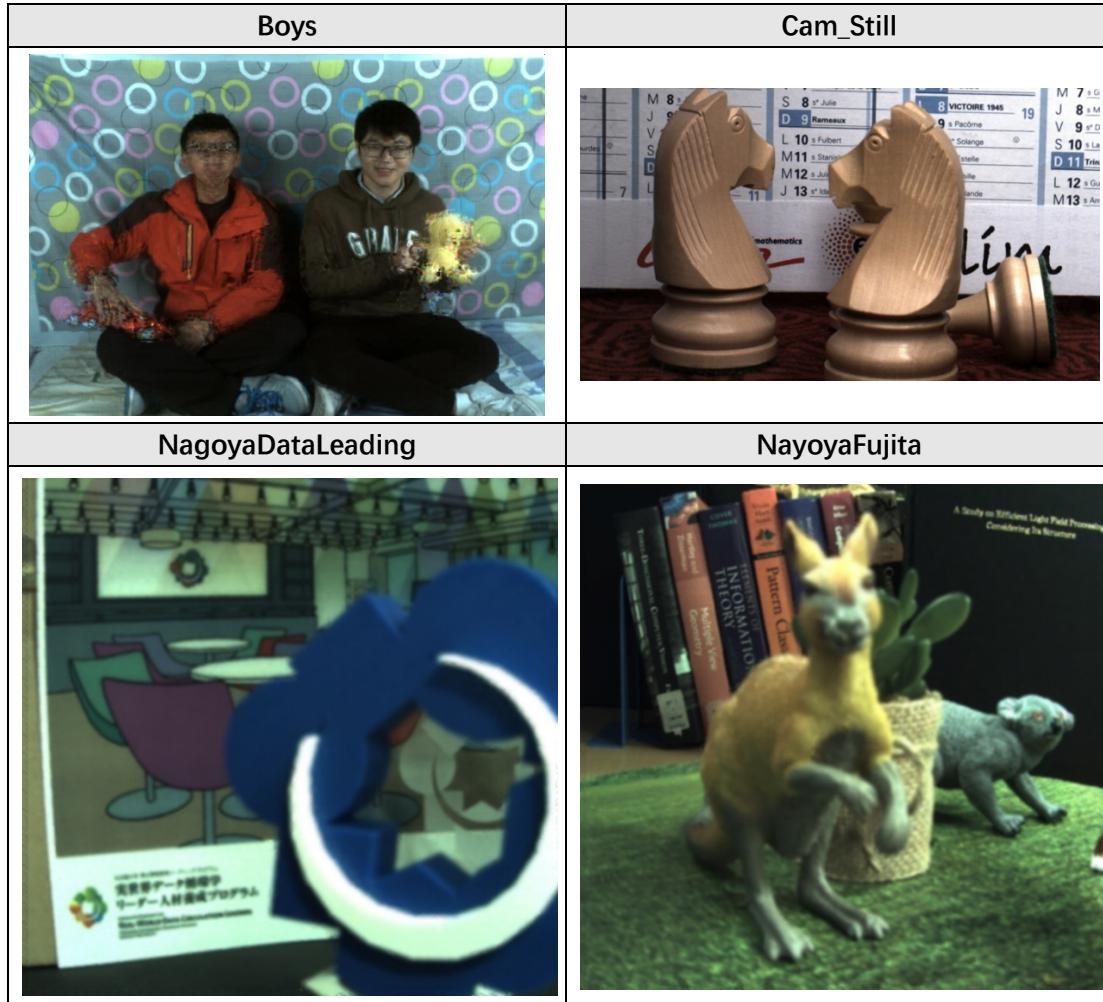
注：baseline 为直接拿上一帧 R5 作为结果与 R14 计算 PSNR 值。

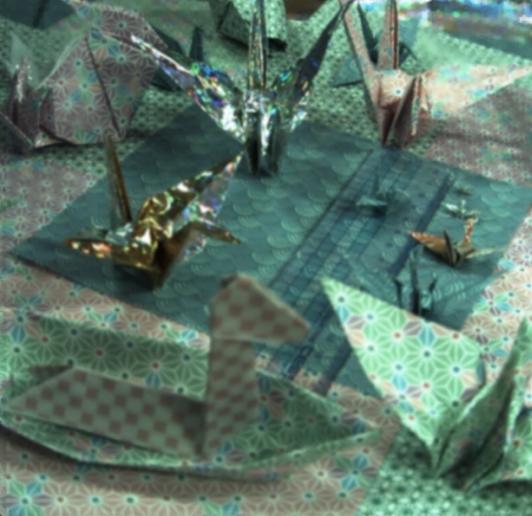
表 5.2 中，weight 模型使用的 block 大小为 8，微位移 pad\_size 量为 4，psnr\_mat thresh 阈值为 37，此时 weight 模型比起 baseline 直接使用上一帧信息均有明显的结果提升。

- 图片结果：

weight 模型在所有的数据上的预测图片结果展示，大图请见一同上传的图片文件。

表 5.4 weight 模型图片结果



NagoyaOrigami	Toys
	
Trees	-
	

weight 模型除了 Boys 数据外，预测的结果肉眼观察均有很好的结果。而 weight 模型在 Boys 数据上暴露了短板，无法处理帧间变动幅度过大且不规则的情形。因此我们设计出了 transformer 模型。

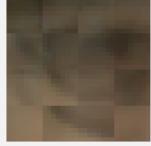
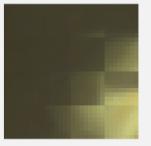
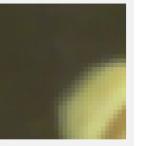
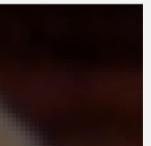
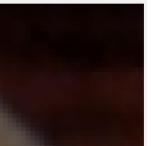
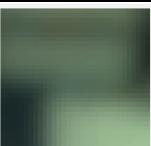
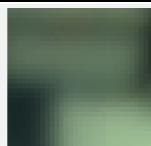
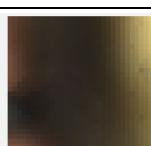
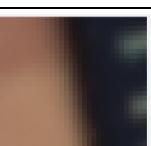
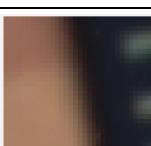
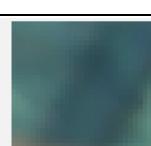
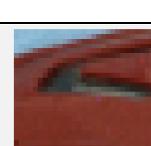
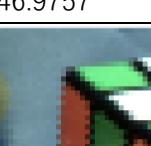
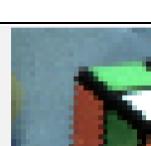
### 5.3 其他模型

模型改进过程中的中间模型的部分结果展示。

#### 5.3.1 简单权重模型

从每个数据集中取前后帧变化最大的块展示小部分结果

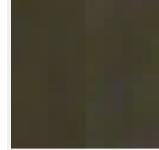
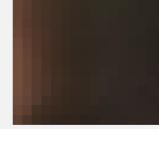
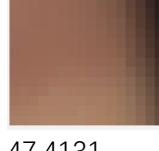
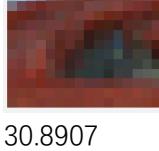
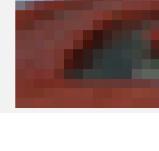
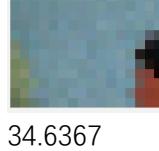
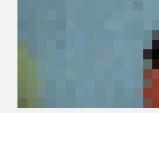
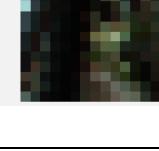
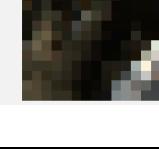
表 5.5 简单权重部分结果

数据集	部分结果 1 (左 : 预测, 右 : 真实)		部分结果 2 (左 : 预测, 右 : 真实)	
Boys				
	24.0143		22.0718	
Cam_still				
	32.5395		49.8327	
Leading				
	49.1222		48.5474	
Fujita				
	46.5302		49.1885	
Origami				
	39.2129		46.9757	
Toys				
	40.0239		35.3245	
Trees				
	35.3610		35.1817	

从上面结果可以看出，简单权重模型对帧间变化不大的情形能有较好的结果，但前后帧变化幅度大且不规则时，如 Boys 的结果，就会出现预测失误；此外简单权重模型可以较好的应对前后平移变换的场景 Trees，甚至有比复杂权重模型 weight 有更好的结果。

### 5.3.2 矩阵平移变换模型 v1

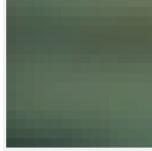
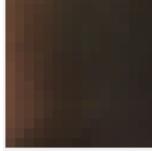
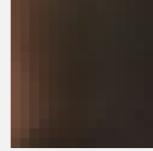
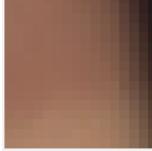
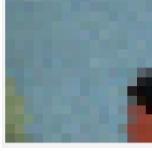
表 5.6 矩阵平移变换模型 v1 部分结果

数据集	部分结果 1 (左：预测，右：真实)		部分结果 2 (左：预测，右：真实)	
Boys				
	24.2377		42.2060	
Cam_still				
	44.8069		49.9172	
Leading				
	50.4716		49.3352	
Fujita				
	44.6621		47.4131	
Origami				
	45.8183		49.2685	
Toys				
	30.8907		34.6367	
Trees				
	15.2040		15.3970	

可见矩阵平移变化模型的左右变换相同假设、左右分开预测的方式并不成立，一旦场景由波动，就会出现明显的预测错误。

### 5.3.3 矩阵平移变换模型 v2

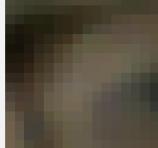
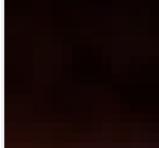
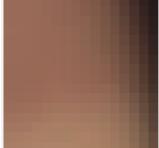
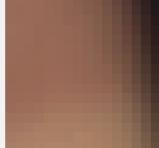
表 5.7 矩阵平移变换模型 v2 部分结果

数据集	部分结果 1 (左 : 预测, 右 : 真实)		部分结果 2 (左 : 预测, 右 : 真实)	
Boys				
	24.2377		42.2060	
Cam_still				
	44.8069		49.9172	
Leading				
	50.4716		49.3352	
Fujita				
	44.6621		47.4131	
Origami				
	45.8183		49.2685	
Toys				
	30.8907		34.6367	
Trees				
	15.2040		15.3970	

v2 相比于 v1 只是改善了内存消耗量和计算复杂度，并没有改善 v1 的结果，所有的计算结果与 v1 相同。

### 5.3.4 帧间时域变化模型

表 5.8 帧间时域变化模型部分结果

数据集	部分结果 1 (左 : 预测, 右 : 真实)		部分结果 2 (左 : 预测, 右 : 真实)	
Boys				
	34.8826		27.7505	
Cam_still				
	40.5558		53.9959	
Leading				
	52.1988		51.2095	
Fujita				
	47.6243		49.1027	
Origami				
	37.3892		49.9002	
Toys				
	38.6123		35.6763	
Trees				
	23.6705		27.0611	

结果证明帧间时域的变换的假设基本合理, 仅仅引入了一小部分的噪声就解决了 weight 模型在 boys 数据集上的模糊问题 (人总算有人样了), 但是依然存在模糊和离散现象。

## 6 对比实验

### 6.1 transformer 与 weight 结果比较

transformer 与 weight 模型的 PSNR 结果比较如下所示：

表 6.1 transformer 与 weight 模型 PSNR 结果表

model	Boys	Cam_still	Leading	Fujita	Origami	Toys	Trees	Average
transformer <sup>1</sup>	42.2212	41.0064	52.6149	52.7888	49.8917	35.2635	28.2809	43.1525
transformer <sup>2</sup>	<b>42.2212</b>	<b>41.0174</b>	<b>52.6149</b>	<b>52.7888</b>	<b>49.8917</b>	37.5560	28.2809	<b>43.4816</b>
weight	29.1528	38.3948	46.1306	44.6220	44.0467	<b>37.9978</b>	<b>28.8066</b>	38.4502
baseline	22.0778	24.2851	45.9011	38.2991	43.2356	30.8380	12.2819	30.9884

注：transformer1：计算并优化全部的块；transformer2：每张图选取最优的 thresh；weight：psnr\_mat thresh=37。baseline 为直接拿上一帧 R5 作为结果与 R14 计算 PSNR 值。

由表 6.1 可得，transformer 模型在各种数据集上均有较为突出的结果，而 weight 模型对于前后帧变动幅度大的场景无法做到很好的处理，尤其是 Boys 数据集，此外在容易预测的 Leading、Fujita、Origami、和 Toys 场景，weight 与 transformer 都有很高且接近的 psnr 值。

### 6.2 微位移方法对比实验

#### 6.2.1 微位移对 transformer 模型的影响

为了方便比较，我们取 block\_size=16，并且 transformer 模型不使用下一帧的信息，在 Boys 数据集的左侧男孩脸部进行比较微位移量大小对 transformer 的影响：

微位移量	图片结果		PSNR
0			32.3728
1			36.5945
2			37.1718

由上面结果可知，不使用微位移时，人眼图片出现大量的噪点，而一旦开始使用微位移，

位移量为 1 时，就让噪点几乎消失，而随着位移量的不断增大，psnr 值不断上升，证明了微位移方法能见效模型的不稳定性，减少最优集的范围，使结果噪点更少。

### 6.2.2 微位移对 weight 模型的影响

为了方便比较，我们取 block\_size=8，在 Trees 数据集的一小部分上进行比较微位移量大小对 transformer 的影响：

微位移量	图片结果		PSNR
0			26.1512
1			29.8510
2			32.5651
3			33.7536
4			34.7543

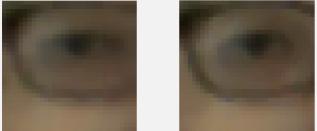
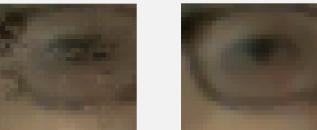
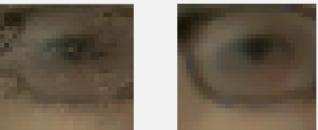
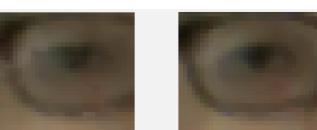
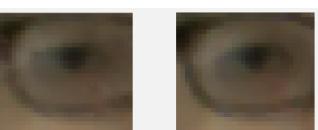
由结果可见，在 weight 模型上，微位移同样起到了极大的作用。不使用微位移时，图片出现大量的噪点，而一旦开始使用微位移，位移量为 1 时，就让噪点几乎消失，而随着位移量的不断增大，psnr 值不断上升。

weight 与 transformer 模型上的结果证明了微位移方法不仅能使图像噪点更少，还具有模型通用性。

### 6.3 transformer 块大小参数与微位移量的选取

在选取 transformer 块大小的过程中，为了便于实验，我们将 transformer 设置成只使用上一帧信息来减少计算量、同时突出 block 大小、微位移量大小对模型结果的影响。

block size	微位移量	结果：图片、PSNR、time/pixel	block size	微位移量	结果：图片、PSNR、time/pixel
8	0	 29.2271, 0.017s	10	0	 30.2693, 0.013s
8	1	 31.1520, 0.0625s	10	1	 32.3694, 0.05s
8	2	 31.8260, 0.156s	10	2	 33.2559, 0.15s
8	3	 32.0266, 0.3083s	10	3	 33.6138, 0.300s
12	0	 32.2837, 0.0117s	16	0	 32.3728, 0.0146s
12	1	 35.4945, 0.0506s	16	1	 36.5945, 0.0615s
12	2	 35.9830, 0.1369s	16	2	 37.1718, 0.1844s

12	3		-	-		
		35.9671, 0.3214s				
20	0		32	0		31.2309, 0.0413s
		32.7992, 0.0182s				
20	1		32	1		36.6696, 0.1560s
		36.6824, 0.0697s				
20	2		32	2		37.2337, 0.4825s
		37.0509, 0.2086s				

从上对比实验可见，当 block\_size 从 8 增加到 16 时，预测的结果有显著的提升，而时间的消耗只呈现小幅度上升；而从 16 增加到 32 的过程中，不仅预测结果没有提升，时间消耗甚至成幂次增加，计算开销更大。而对于微位移量，从 0 增加到 2 的过程都会使得 PSNR 值快速提升，而 2 增加到 3 则只有小幅度提升，但是时间开销增加到原先的 4 倍。

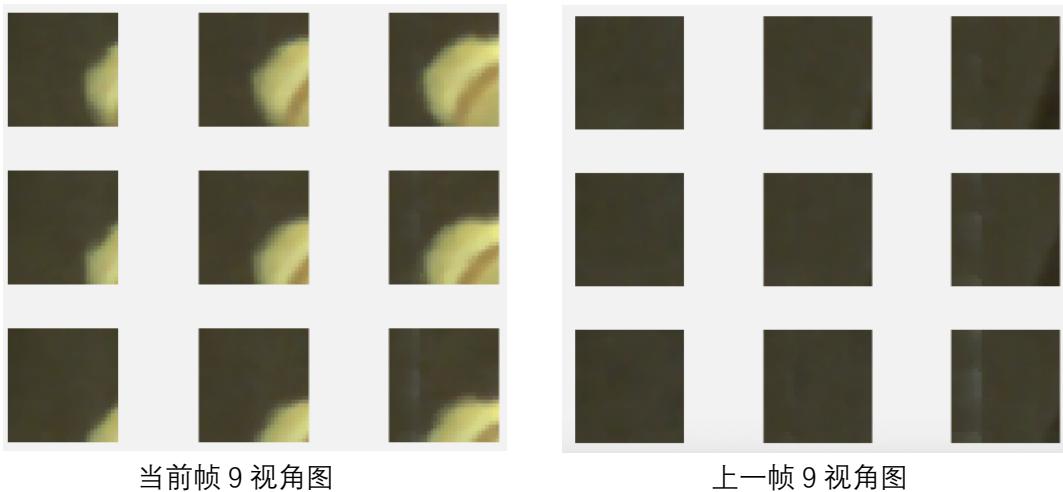
因此在权衡 PSNR 效果和计算时间的考虑下，我们最终在使用当前帧的 transformer 模型上设置的 block 大小为 16，微位移量为 1。

## 6.4 transformer 有无使用当前帧对比实验

最开始我们的 transformer 模型仅仅使用了上一帧 $\langle R1, R2, R3 \rangle < R4, R5, R6 \rangle$ 、 $\langle R7, R8, R9 \rangle$ ，并没有使用下一帧。在 Boys 数据集上出现了如下问题：



玩偶的边缘部分出现了严重的离散与模糊的错误情况。于是我们分析了前后两帧在该块上 9 个视角的图：



可以看到上一帧中每个视角的图片几乎都为纯色，模型从上一帧中难以得到平移变换等信息。考虑到这一点，我们添加了当前帧 $\langle R10, R11, R12 \rangle$ 、 $\langle R16, R17, R18 \rangle$ 的数据（不包括目标所在行 $\langle R13, R14, R15 \rangle$ ），让模型的变换同时满足当前帧的这两组数据，使得结果有了明显的提升。

不使用当前帧信息	使用当前帧信息
	
PSNR: 26.4040	PSNR: 40.2792

可见一旦使用当前帧的信息，就能弥补上一帧视差信息不全面的问题，使得结果有极大的提升，视差变换矩阵的求解变得更为精确。

## 6.5 PSNR 近似矩阵阈值对 transformer 结果的影响

PSNR 矩阵阈值的大小对 transformer 模型在各个数据上结果的影响统计下表所示：

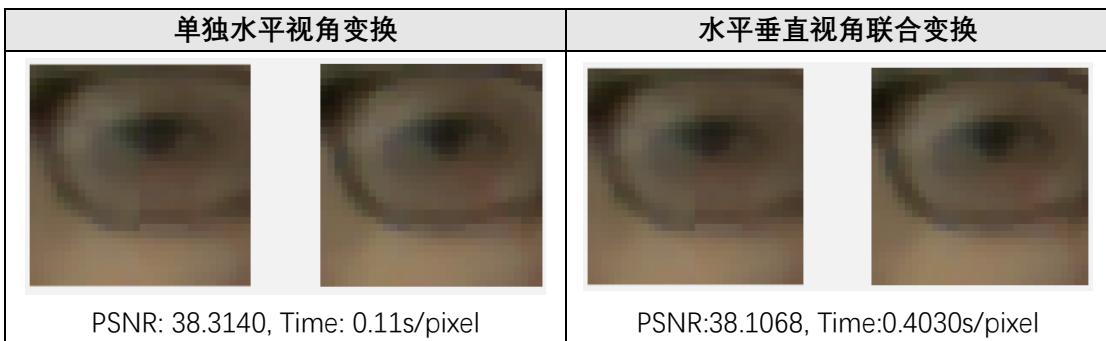
表 6.2 PSNR 矩阵阈值对 transformer 性能影响统计表

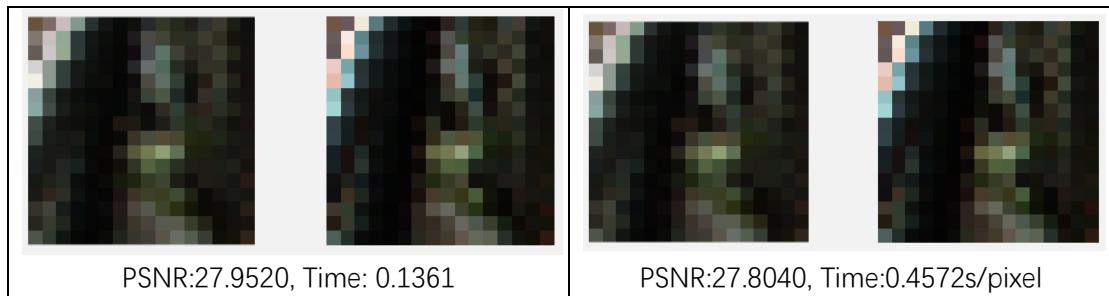
thresh	Boys	Cam_still	Leading	Fujita	Origami	Toys	Trees
<b>all_block</b>	<b>42.2212</b>	41.0064	<b>52.6149</b>	<b>52.7888</b>	<b>49.8917</b>	35.2635	28.2809
<b>50</b>	42.2212	41.0063	52.5153	52.2824	49.8830	35.2636	28.2809
<b>49</b>	42.2208	41.0063	51.8163	51.6856	49.7867	35.2636	28.2809
<b>48</b>	42.2184	41.0063	50.4127	51.0484	49.4183	35.2637	28.2809
<b>47</b>	42.2025	41.0063	49.1743	50.3626	48.8361	35.2637	28.2809
<b>46</b>	42.1552	41.0061	48.4674	49.6553	48.1133	35.2641	28.2809
<b>45</b>	42.1026	41.0067	47.8831	48.9849	47.4097	35.2644	28.2809
<b>44</b>	42.0756	<b>41.0174</b>	47.4195	48.3308	46.6945	35.2686	28.2809
<b>43</b>	42.0420	41.0092	46.9586	47.8053	46.0690	35.3061	28.2809
<b>42</b>	42.0004	40.9979	46.6430	47.2881	45.4734	35.3580	28.2809
<b>41</b>	41.9077	40.9798	46.3950	46.6593	45.0185	35.4161	28.2809
<b>40</b>	41.5771	40.9337	46.2546	46.2281	44.6352	35.6383	28.2809
<b>39</b>	41.1194	40.9000	46.1554	45.6169	44.2943	36.1680	28.2809
<b>38</b>	40.7700	40.8310	46.1015	45.1012	44.0611	36.6504	28.2809
<b>37</b>	40.5745	40.6035	46.0884	44.5508	43.8016	37.1762	28.2809
<b>36</b>	40.4071	40.1188	46.0808	44.0077	43.6197	<b>37.5560</b>	28.2809
<b>35</b>	40.2501	39.4692	46.0532	43.4430	43.4957	37.5330	28.2809

表 6.2 中 all\_block 代表对所有的块进行优化。对所有的块进行优化时，PSNR 的性能基本为最优，但是在 Toys 图片集上当设置阈值为 36 时性能最优。Trees 图片集由于求得的 PSNR 近似矩阵的 PSNR 都极低，因此表中的阈值都会使其优化全部的块，结果相同。

## 6.6 transformer 有无水平垂直视角联合的结果对比

transformer 模型可以进一步地将水平变换与垂直变换相互联合，为了研究联合变换是否能提升模型的结果，我们在数据集的一小部分上做了对比实验。





从结果中可以看出，联合变换并不能提升结果，甚至引入了更多的噪声，而且时间复杂度为单独水平变换的 4 倍，并不能给 transformer 模型带来更多的提升。因此我们的 transformer 采用单独的水平变换而不是联合变换。

## 7 总结与展望

这次实验的过程比较坎坷，一开始用根据视角变换设计的矩阵平移变换模型在复杂场景会出现光栅状的结果，而之后设计的 weight 模型一开始出现了结果不稳定的情况，为了解决这一问题我们提出了微位移的方法，最后 weight 模型在大部分的场景都跑出了很优秀的结果，但是在 Boys 图片上出现了马赛克。

为了解决 weight 模型的问题，最开始我们提出了时域上的 transformer，但是时域上的变化没有准确的监督信息，只能用周边 8 领域视角来做近似，因此在 Boys 图片集还是出现了模糊现象。后来突然想到可以把平移变换假设和 weight 结合起来，在前一帧的空域上来学习视角的变换，继而设计出了 transformer 模型。而 transformer 模型在 boys 数据集上效果虽然有很大的提升，但是还是出现部分模糊问题。后来分析了前一帧的 9 视角的图片发现前一帧几乎为纯色、视角信息很少，这才导致了视角变换求解失常。后来灵机一动，想到当前帧的视角变换信息还没用上，加上之后就完全解决了 Boys 数据的问题，使其 psnr 结果上升到了 42.2212。

此外，目前我们的模型是对大部分的块进行了优化求解，但实际上相邻的块的变化矩阵应该极为相似，我们之后将基于这一点对模型进行改良，通过求解部分变化矩阵来实现全部块的变化，以节省计算时间。

在项目实现过程中，我们努力尝试着使用凸优化的理论，只使用凸优化方法通过 cvx 求解器实现了 weight 和 transformer 两个模型，我们还用凸优化理论去推导证明微位移方法的可行性，而且实际的结果证明了微位移方法确实减小了最优集，让解的结果更为准确，体验了一波理论推导在实际起作用的爽快感。

而对于未来可以提升的方向，我们将研究如何减少模型计算量的方法以及提升模型的预测结果。目前我们得出了几个可行的方向，一个是对多个块（相同列）甚至是全图使用同一个块计算得出的变换矩阵  $r2l$  和  $I2r$ （因为在同一列上、或是相邻的块其水平视角的变换非常相似，对每一个变换相似的块进行求解浪费了计算资源）。另一个可行的方向是可以尝试将图片下采样后求解变换矩阵，然后设计相应的方法把求得的变换矩阵“上采样”到原始图片尺寸的范围，再进行大的块求解。最后是考虑结合图像处理的方法，对原始图片进行预处理（上采样、降噪等方法），使得我们的模型能求解出更为准确的视角变换矩阵。

总的来说，我们提出了微位移方法来减小最优集的范围，使模型求解的结果更为准确，并且我们从权重和视角变换两个角度设计了 weight 和 transformer 两个模型，通过矩阵来实现相应的变换，仅使用凸优化的方法就获得了极高的 PSNR 值和视觉效果良好的预测图片。