《凸优化》课程大作业报告

Inter-prediction for Multiview video coding

（多视图视频编码的帧间预测）

班级：深海洋硕 19 班 深数据硕 19 班

姓名：杨毅远 谭世琦

学号：2019211330 2019211313

时间：2019 年 12 月 19 日

目录

一、 实验目的与要求

在多视角视频领域，对于多角度的视频的大数据量通常需要压缩。在对于高帧率视频的压缩中，我们通常会遇到如下问题，即对于前后相邻的两帧图像，因为十分相似，所以会有许多的冗余(重复性)信息；除此之外，对于同一帧但不同拍摄角度的多幅图像，由于摄像头的角度变动范围较小，所以通常也会存在众多的冗余（重复性）信息。所以在图像压缩中，我们会对于以上的问题进行分析，从而进一步压缩视频的存储规模。

在本实验中，我们需要根据前后相邻两帧的共 17 幅图像，进行其中 1 幅图像的预测。其中 17 幅图像分别为，前一帧图像的 9 个不同角度位置拍摄的图片，其中编号依次为左上角的图片为 R1，上部中间的图片为 R2，右上角的图片为 R3，右侧中间的图片编号为 R4，中心图片编号为 R5，以此类推；对于后 8 个图像是当前帧的 8 幅图片，相比于前一帧的 9 张图片，缺少的中心的图像即是我们要预

测的图像。上述过程可以用下图 1.1 表示：

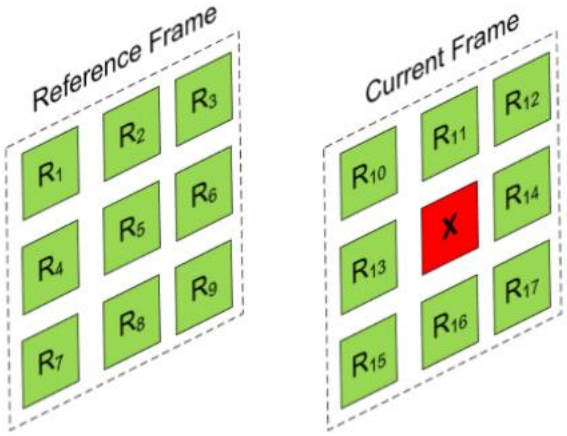


图 1.1　实验问题描述

上图中的 X 就是我们最终需要预测的图像。此问题可以归纳为根据参考图像 R1-R17 生成预测的图像，并尽可能使得预测图像 X 与 label 的 PSNR 尽可能的大。此问题的数学表示如下：

$$\hat{X} = f(R_1, R_2, R_3 \dots R_{17})$$

$$wish\ Min.Diff(X, \hat{X}) \quad \text{or} \quad Max.PSNR(X, \hat{X})$$

对于数据集，我们要用到的数据集包括以下七组图像：Boys、Cam_Still、NagoyaDataLeading、NagoyaFujita、NagoyaOrigami、Toys 和 Trees。其都为 RGB 三通道，范围为 0-255 的图像。

实验要求，图像 X 的真实值不能在计算中出现；在算法中必须使用前一帧图像中的至少一幅图像，必须使用当前帧中的至少一幅图像进行预测；对于图像要进行分块化处理，而且每一个小图像块的区域大小需要在以下范围内选择：4×4，8×8，16×16，32×32，64×64，同时要保证 R1-R17 的分割大小与预测图像 X 的分割大小相同，且一一匹配。此过程可以用图 1.2 进行表示：

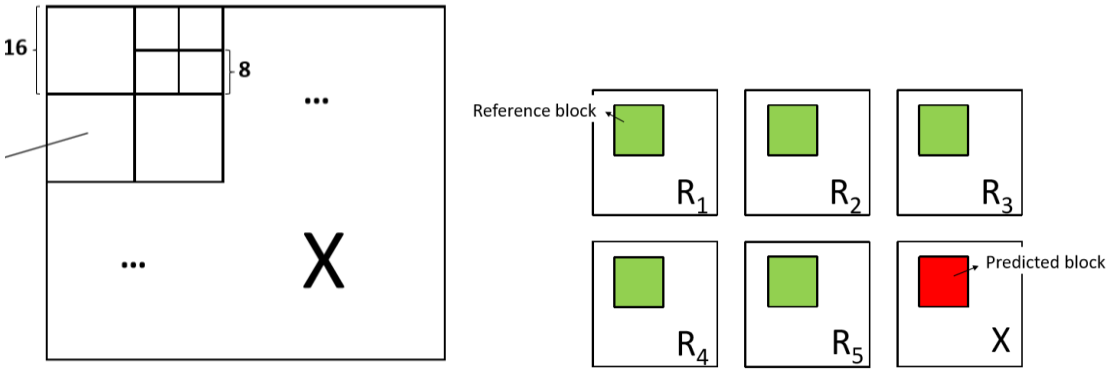


图 1.2　分割的问题直观描述

最终的计算指标是 RBG 三通道的 RSNR 的平均值，具体计算公式如下：

$$PSNR_{mean} = \left(PSNR\left(X_R, \widehat{X_R}\right) + PSNR\left(X_G, \widehat{X_G}\right) + PSNR\left(X_B, \widehat{X_B}\right)\right)/3$$

二、　实验过程与具体算法

2.1 算法流程简单描述

我们首先将此优化问题归纳为以下的步骤：

1. 将 R1-R17 这 17 幅图片分割。分割的区域的边长应符合题目要求，并将分割后的区域打上标签$Rx(i,j)$，$Rx(i,j)$表示在图x中水平方向的第i个，在垂直方向的第j个 block，这样可以做到 17 个图像中的小区域的一一对应。
2. 在前一帧的 R1-R9 中得到分割后的每一个小区域中$R2(i,j)$、$R4(i,j)$、$R6(i,j)$、$R8(i,j)$与$R5(i,j)$的对应关系。此问题可以表示为下面的数学问题：

$for\ i\ in\ range(block_{num\ row})$:

$for\ j\ in\ range(block_{num\ col})$:

$$\widehat{R5}(i,j) = \omega_1(i,j) * R2(i,j) + \omega_2(i,j) * R4(i,j) + \omega_3(i,j) * R6(i,j) + \omega_4(i,j) * R8(i,j)$$

其中$\omega_x(i,j)$是位置为i，j的权重矩阵。

使用 CVX，设置权重矩阵$\omega_x(i,j)$作为优化变量，$min.diff\left(R5,\widehat{R5}\right)$为优化目标，进行优化得到最优的权重矩阵$\omega_{xbest}(i,j)$。

3. 将第（2）步得到最优的权重矩阵$\omega_{xbest}(i,j)$进行直接使用到当前帧的图片中并使用此计算出$\hat{X}$。

$for\ i\ in\ range(block_{num\ row})$:

$for\ j\ in\ range(block_{num\ col})$:

$$\hat{X}(i,j)=\omega_{1best}(i,j)*R11(i,j)+\omega_{2best}(i,j)*R13(i,j)+\omega_{3best}(i,j)*R14(i,j)+\omega_{4best}(i,j)*R16(i,j)$$

并计算最终的 PSNR 指标。

整体算法的主要伪代码如下：

$for\ i\ in\ range(block_{num\ row})$:

$for\ j\ in\ range(block_{num\ col})$:

$var\ W=[w_1,w_2,w_3,w_4]=blk_{size}*blk_{size}*4$

$\text{minimize norm}(R5(i,j),\ \omega_1*R2(i,j)+\omega_2*R4(i,j)+\omega_3*R6(i,j)+\omega_4*R8(i,j))$

$\hat{X}(i,j)=\omega_{1best}*R11(i,j)+\omega_{2best}*R13(i,j)+\omega_{3best}*R14(i,j)+\omega_{4best}*R16(i,j)$

$PSNR(X,\hat{X})$

算法的时间复杂度计算公式如下：

$$Complexity=lock_{num\ row}{*}block_{num\ col}{*}\ complexity\_of\_cvx\_problem$$

整体的算法流程如下图所示，下面将详细介绍每一步骤的具体实现流程，以及步骤内对于问题的分类讨论情况：

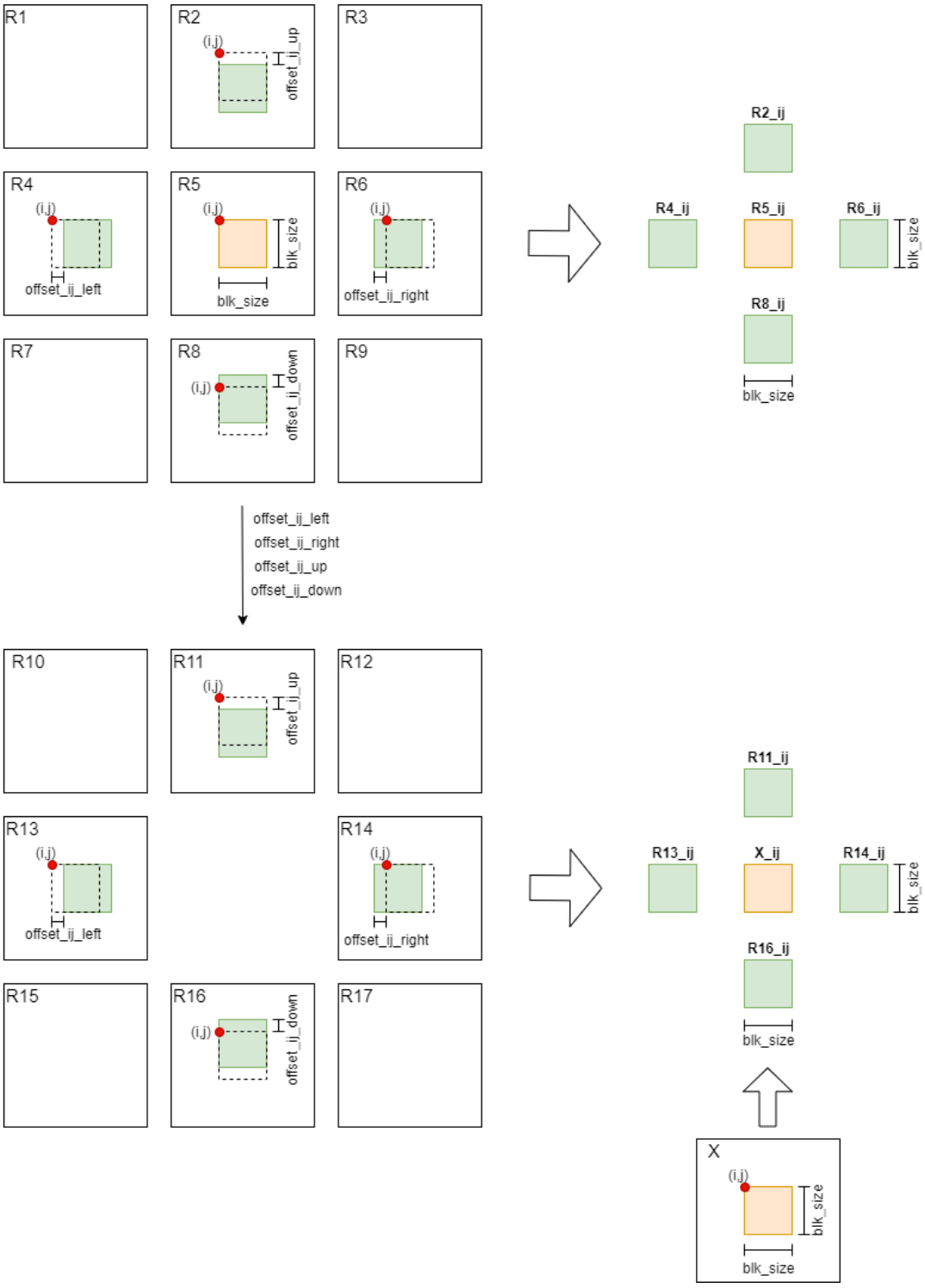


图 2.1　整体算法的流程图

2.2 算法每一步骤的具体实现

(1) 在切割步骤中，由于实验对于 block 的边长有约束，所以这里使用了求模的方法，对于不够 block 边长的区域进行边界补零操作。并将分割后的结果打

上标签：$Rx(i,j)$，即表示在图x中水平方向的第i个、在垂直方向的第j个 block。

(2) 计算在前一帧的图像中分割后的每一个小区域中$R2(i,j)$、$R4(i,j)$、$R6(i,j)$、$R8(i,j)$与$R5(i,j)$的对应关系。在这里考虑到摄像机不同的移动方向，我们对于实际的 block 进行了位置的调整，例如$R2(i,j)$是$R5(i,j)$在摄像头“上仰”时得到的图像，所以与$R5(i,j)$的图像最接近的$R2$中的图像应该位于$R2(i,j)$的正下方附近，所以，在这里我们设定了一个 offset，让$R2(i,j)$这个 block 在 offset 中进行步长为 1，方向为下的平移，并在每一次平移后进行新的$\widehat{R2}(i,j)$与$R5(i,j)$的 PSNR 的计算，最终计算得到 PSNR 最大的$\widehat{R2}(i,j)$作为我们的实际算法中使用的$R2(i,j)$这个 block。我们对于 R4、R6 以及 R8 也进行上述操作，与 R2 不同的是，R4 的 offset 方向是向右的；R6 的 offset 方向是向左的；R8 的 offset 方向是向上的，因为这些方向与对应的 R5 的匹配度是最高的。最终得到一个小 block 中的 5 个图像。具体的实验过程可以表示为图 2.2：

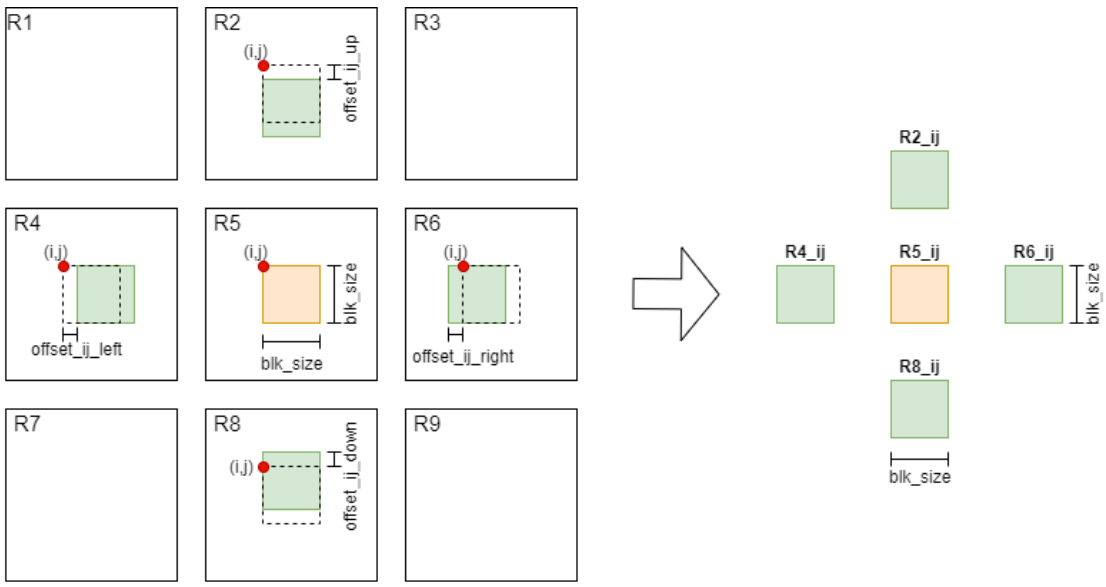


图 2.2　步骤二的可视化过程

在此基础上，我们将计算得到的$R2(i,j)$、$R4(i,j)$、$R6(i,j)$、$R8(i,j)$看作自变量，将$\widehat{R5}(i,j)$视为因变量，进行模型构建，即：

$$\widehat{R5}(i,j) = \omega_1(i,j) * R2(i,j) + \omega_2(i,j) * R4(i,j) + \omega_3(i,j) * R6(i,j) + \omega_4(i,j) * R8(i,j)$$

其中的$\omega_1(i,j)$表示 $R2(i,j)$对应的权重矩阵，大小为$block_size * block_size$的，$R2(i,j)$的大小为$block_{size} * block_{size} * 3$的，这里的 3 指的是 RGB 三通道，这里我们认为同一 pixel 上的 RBG 值的权重相同，这样可以减少优化参数，提高运算速度。

在实验中，我们发现待处理 block 有两种情况，其中一种为前一帧的 block 和对应的当前帧的 block 无较为明显的差异，这种情况下我们选择将每一个 block 中进行 pixel-to-pixel 的匹配，即使用权重矩阵$\omega_x(i,j)$进行模型的优化，这里的权重矩阵就是大小为$block_size * block_size$的矩阵，图形化表示为图 2.3：

图 2.3　基于第一种权重矩阵的表达式

在另一方面，前一帧的 block 和对应的当前帧的 block 有较明显的差异，例如在 boys 图片中，出现了右侧男生手中的玩具熊的位置明显变化的情况，这时再使用精细化的 pixel-to-pixel 的方法没有意义，而且会降低运算速度并造成过拟合。所以在这种情况下，我们采用将权重矩阵$\omega_x(i,j)$变为一个数值，进行简化，其图形化表示为图 2.4：

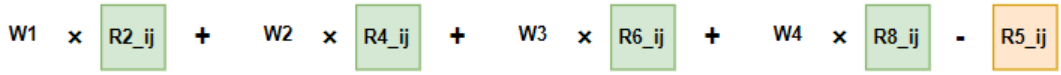


图 2.4　基于第二种权重矩阵的表达式

基于此，构建一个优化函数的模型，如下图 2.5、图 2.6 所示：

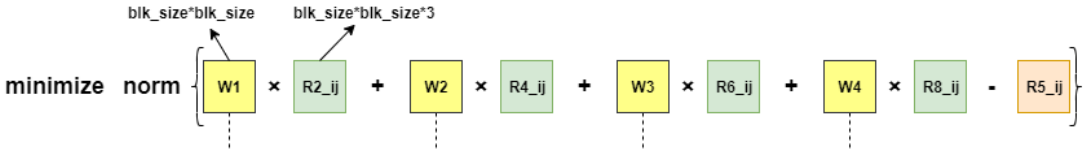


图 2.5　基于第一种权重的优化模型

或者是：

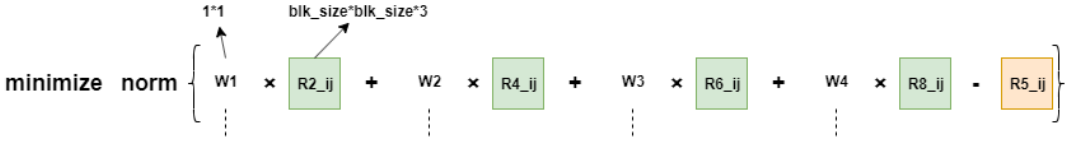


图 2.6　基于第二种权重的优化模型

我们在代码中设置，当前一帧与后一帧对应的 block 的 PSNR 小于某一阈值时（即两 block 不相似），使用基于图 2.4 的权重数值对应的 loss 值进行计算；当前一帧与后一帧对应的 block 的 PSNR 大于某一阈值时（即两 block 相似），使用基于图 2.3 的权重矩阵对应的 loss 值进行计算。此过程的可视化描述见图 2.7：

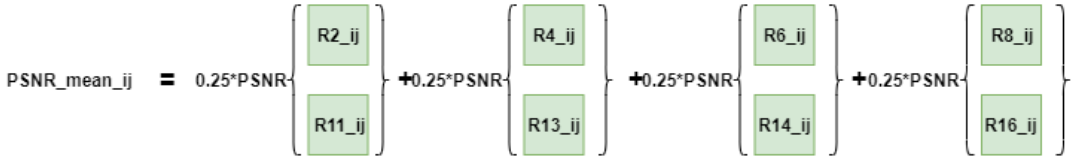


图 2.7　判断使用哪一种权重矩阵的 PSNR 计算过程

除此之外，我们使用了一范数、二范数（速度极慢）、无穷范数以及 Frobenius 范数分别进行优化尝试，得到了最终的最优权重矩阵(权重值)。

(3) 通过上面的步骤，可以得到最优的权重矩阵或者权重参数$\omega_{xbest}(i,j)$，直接使用以下公式进行图像x的计算，计算公式如下。

$$\hat{X}(i,j) = \omega_{1best}(i,j) * R11(i,j) + \omega_{2best}(i,j) * R13(i,j) + \omega_{3best}(i,j) * R14(i,j) + \omega_{4best}(i,j) * R16(i,j)$$

这个过程可以使用图 2.8 进行可视化：

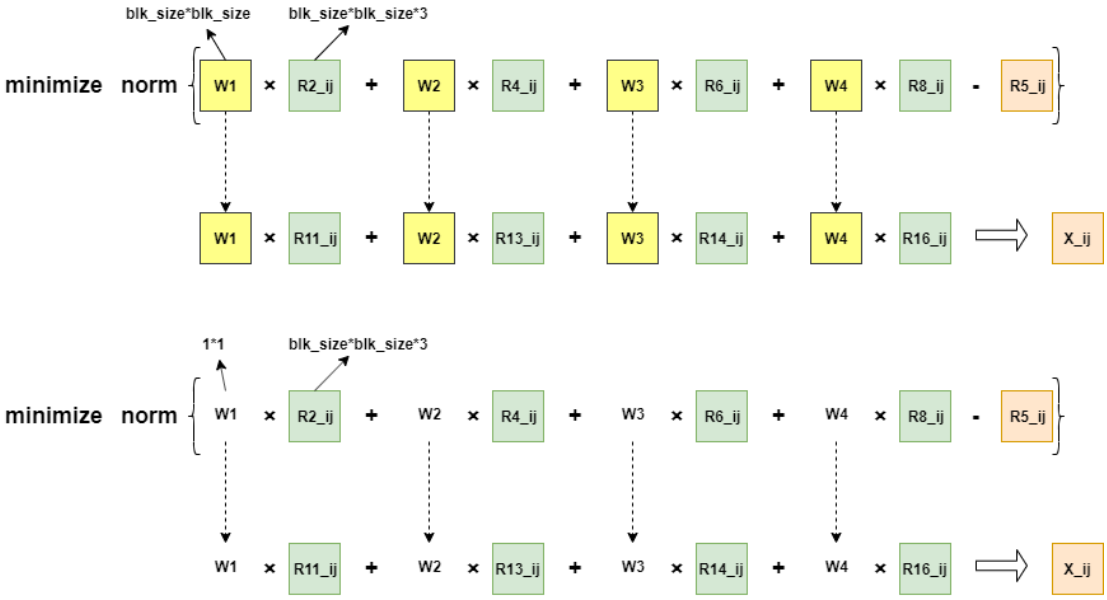


图 2.8　计算预测图像x的可视化过程

计算得到图像x的预测值后，使用 Matlab 自带的 PSNR 函数进行计算，公式如下：

$$PSNR_{mean} = \left(PSNR\left(X_R, \widehat{X_R}\right) + PSNR\left(X_G, \widehat{X_G}\right) + PSNR\left(X_B, \widehat{X_B}\right)\right)/3$$

三、　实验结果与分析

3.1 分块大小对结果的影响

首先我们以 Trees 为例，探究了不同的分块大小对最终结果的影响。当权重ω分别为单值或者矩阵，分块大小分别为 8,16,32 和 64 时，通过上面介绍的方法得到的 Trees 图片的 PSNR 结果如表 3.1 所示：

表 3.1　分块大小对 Trees 预测结果影响表

blk_size	8	16	32	64
$Size(w) = 1 * 1$的 PSNR	28.8400	30.3497	30.4513	30.0822
$Size(w) = blk\_size * blk\_size$ 的 PSNR	26.7190	27.5465	27.6088	27.5115

从以上结果可以看出，无论是当权重ω是单值的时候还是当权重ω是矩阵的时候，均是在$blk_size = 32$的时候取到最优值，对于其他的六组图片的结果也均类似，因此在后续的实验中，均采用 32 为图像分块的大小。

3.2 权重ω对于结果的影响

在$Blk_size = 32$的情况，我们对比了当权重ω是单值和ω是矩阵的情况下，7 组预测图片和原图的 PSNR 对比结果，结果如下表：

表 3.2　w 形态对结果的影响表

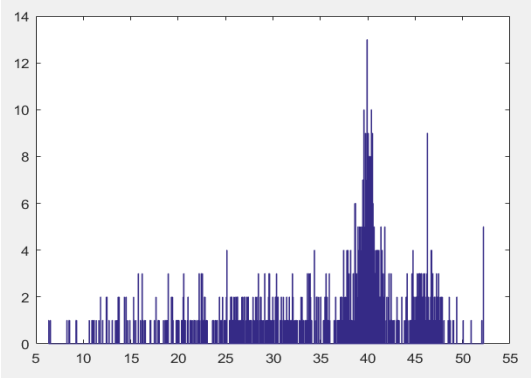
图片名	$Size(w) = 1 * 1$的 PSNR	$Size(w) = blk\_size * blk\_size$ 的 PSNR
Boys	41.8361	44.9631
Cam_Still	42.0561	44.9590
NagoyaDataLeading	47.5145	52.3666
NagoyaFujita	48.1277	51.9861
NagoyaOrigami	44.3422	49.8699
Toys	34.9580	37.5605
Trees	30.4513	27.6088
平均值	41.3266	44.1877

从结果可以看出，对于 Trees 图片，当权重ω的取值是单值时结果的 PSNR 更高，而对于其余的 6 张图片，当权重ω的取值是矩阵的时候，输出图片的 PSNR 更高。这主要是因为，当ω为权重的时候，是在像素级别进行优化，而当ω是单

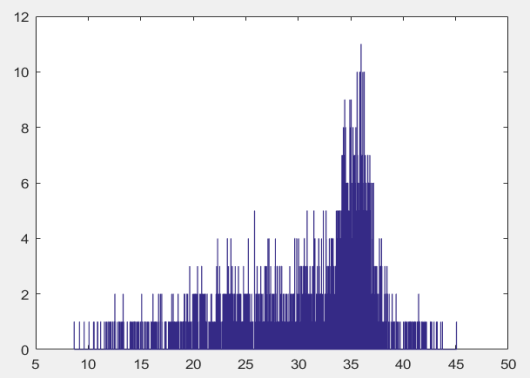
值的时候，是在分块的级别进行的优化。通过观察，发现 Trees 上下两帧图片之间的差别很大，相机存在着轻微的移动，且图片中的树叶会存在一定的移动，因此如果利用上一帧像素级别的权重信息来预测下一帧的图片，可能会产生较大的误差。而对于静态的图片，特别是 NagoyaDataLeading、NagoyaFujita、NagoyaOrigami 这三张图片，上下帧的同一位置的图片的差别很小，但是同一帧相邻位置图片之间的差别较大，且不同纹理部分的差别不同，存在这一定的角度差异，因此仅仅将ω作为一个单值，不能够完全表述存在的这种细节差异，因此当ω为矩阵的时候，最后的结果会比较好。

3.3 模型融合的结果

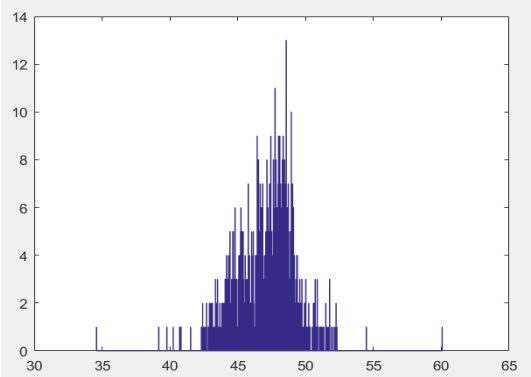
通过观察上面的实验结果，我们发现，Trees 和其他图片的输出结果之间存在这一定的矛盾。为了进一步探索为何会存在这种差别，我们定量化地输出了每张图片的$32 * 32$的分块的$psnr_means$的之别，前面的原理部分已经介绍，$psnr_means$是衡量上下两帧图片的相似性的，下面的每幅图中，横坐标表示$psnr_means$的取值，纵坐标表示对应的分块数量。



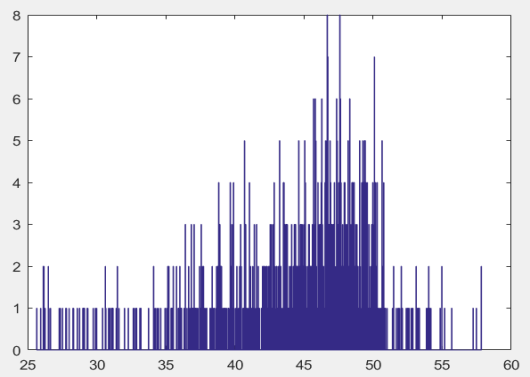
(a) Boys



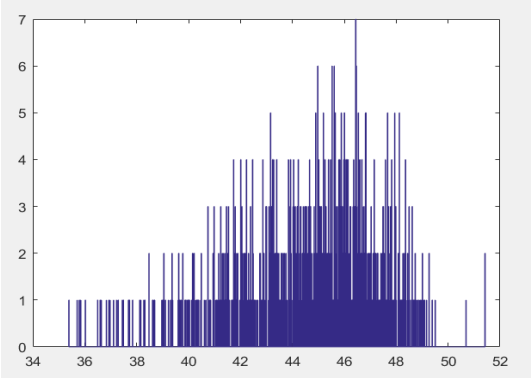
(b) Cam_Still



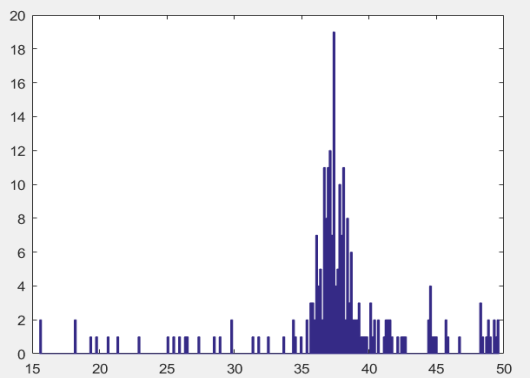
(c) NagoyaDataLeading



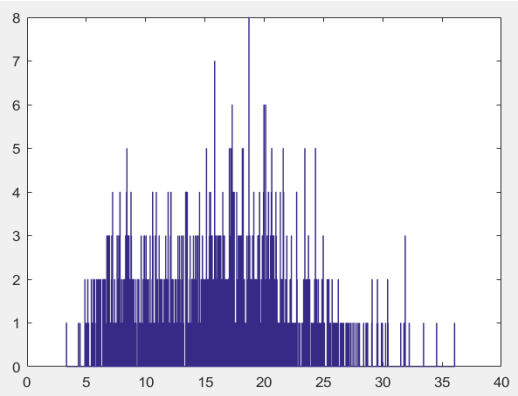
(d) NagoyaFujita



(e) NagoyaOrigami



(f) Toys



(g) Trees

图 3.1　不同图片的分块的前后两帧的$PSNR_means$分布图

从上面的图片可以看出，除了 Trees 图片，其余六幅图的$psnr_means$分布的峰值基本都在 35 以上，而 Trees 的$psnr_means$分布的峰值在 15 左右，这可能是造成 Trees 结果和其余图片结果不同的一个原因，因此我们对于不同$psnr_means$的块进行不同的处理。对于$psnr_means$小于一定阈值的分块，采取的权重ω是一

个单值；对于$psnr_means$超过一定阈值的分块，采取的权重ω是一个矩阵。

通过观察上面$psnr_means$的分布图，我们发现选择 25 作为阈值，可以不影响 NagoyaDataLeading、NagoyaFujita 和 NagoyaOrigami 三张静态的图片，同时能够包含 trees 大部分的分块。最终得到的结果如下表：

表 3.3　阈值方法得到的七个图片的 PSNR 结果

图片名	PSNR 结果
Boys	44.5421
Cam_Still	45.3155
NagoyaDataLeading	52.3666
NagoyaFujita	51.9861
NagoyaOrigami	49.8699
Toys	37.9967
Trees	30.4405

我们将三种方法的结果整合得到如下表格：

表 3.4　三种结果对比表

图片名	$Size(w) = 1 * 1$ 的 PSNR	$Size(w) = blk_size * blk_size$的 PSNR	融合方法 PSNR
Boys	41.8361	44.9631	44.5421
Cam_Still	42.0561	44.9590	45.3155
NagoyaDataLeading	47.5145	52.3666	52.3666
NagoyaFujita	48.1277	51.9861	51.9861
NagoyaOrigami	44.3422	49.8699	49.8699
Toys	34.9580	37.5605	37.9967
Trees	30.4513	27.6088	30.4405
平均值	41.3266	44.1877	44.6453

可以看出，通过将两种模型进行融合，最终得到的结果中，Trees 的结果相对于ω是矩阵的情况，得到了明显的提升，虽然相对于ω是单值的时候还有一点点差距，而且其余的图片的 PSNR 相对于ω是矩阵的时候均变化不大，略有上升或下降，或者保持不变。除此之外，进行模型融合后七组照片的平均 PSNR 得到了较为显著的提高。所以根据图片的先验特性，将两种模型进行融合，能最大程度地合并两种模型的优点，从而达到较好的效果。

最终的结果图如下：

(a) Boys 原图

(b) Boys 预测图

(c) Cam_Still 原图

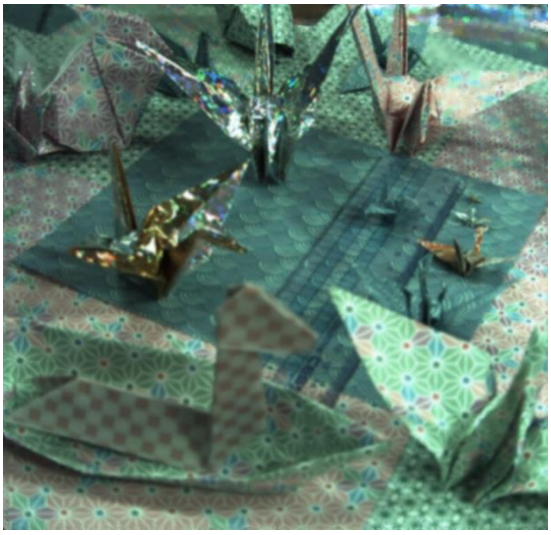
(d) Cam_Still 预测图

(e) NagoyaDataLeading 原图

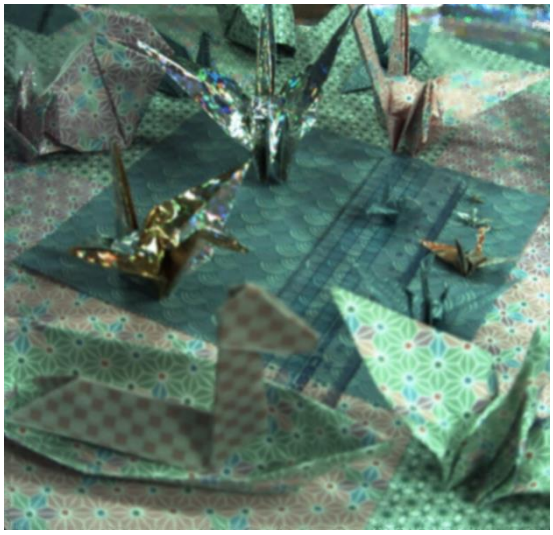
(f) NagoyaDataLeading 预测图

(g) NagoyaFujita 原图

(h) NagoyaFujita 预测图



(i) NagoyaOrigami 原图



(j) NagoyaOrigami 预测图

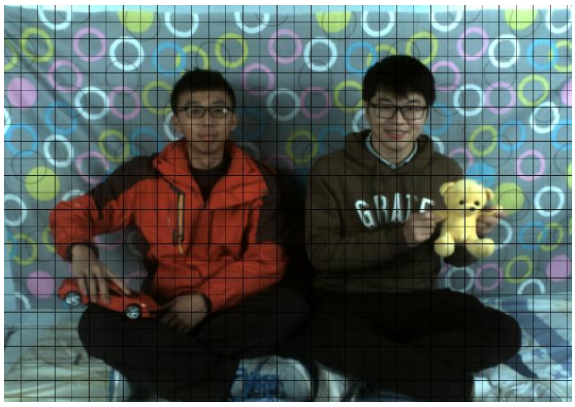
(k) Toys 原图

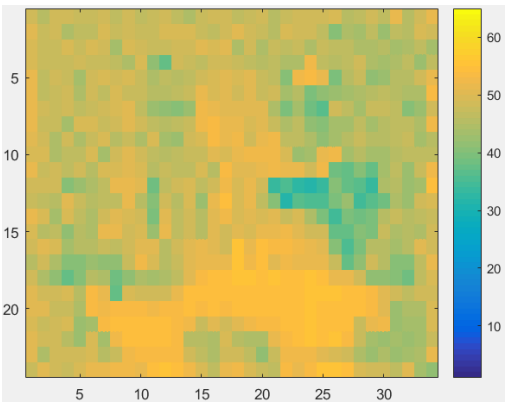
(l) Toys 预测图

(m) Trees 原图

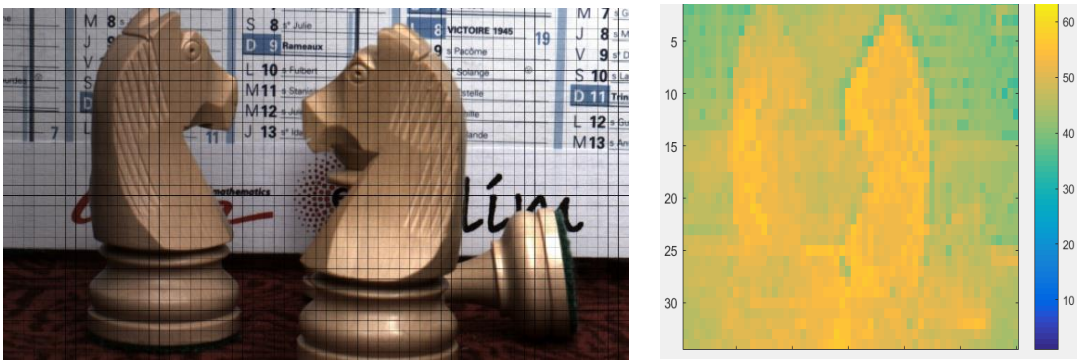
(n) Trees 预测图

各张图片分块的结果及其对应的各个 block 的 PSNR 热力图如下：

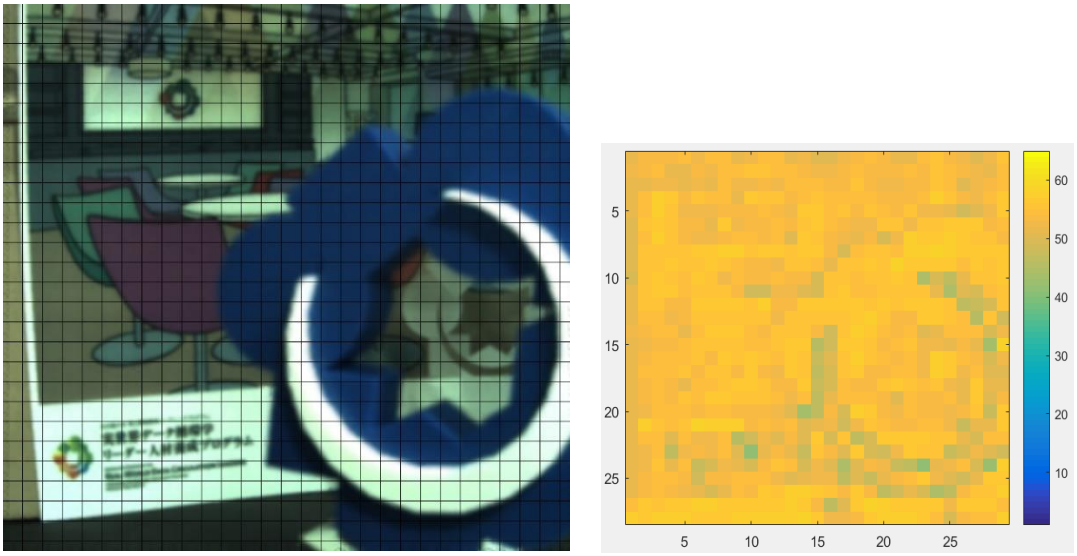




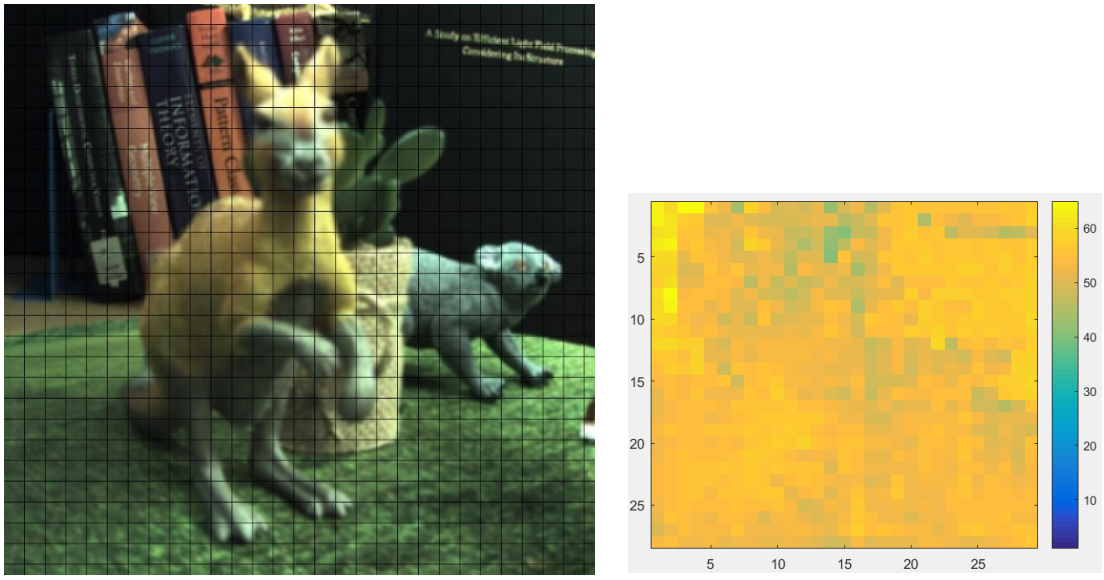
Boys_blocks_num=816



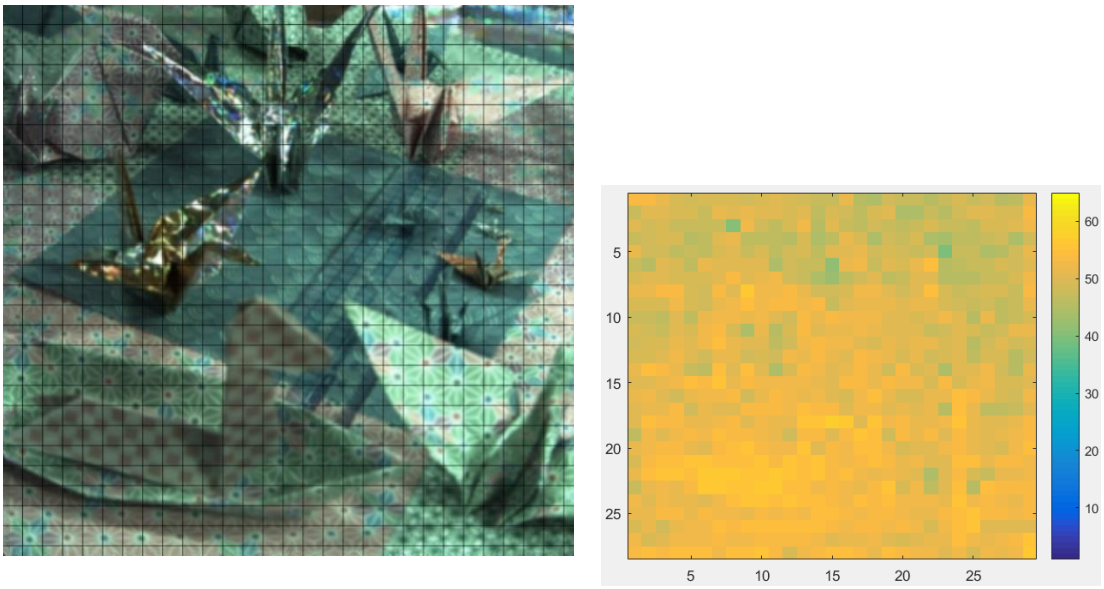
Cam_Still_blocks_num=2040



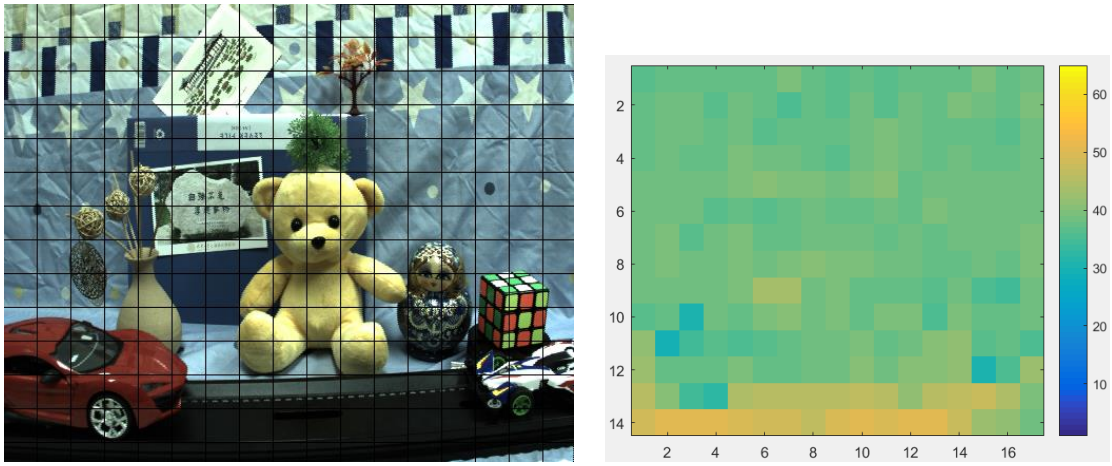
NagoyaDataLeading_blocks_num=812



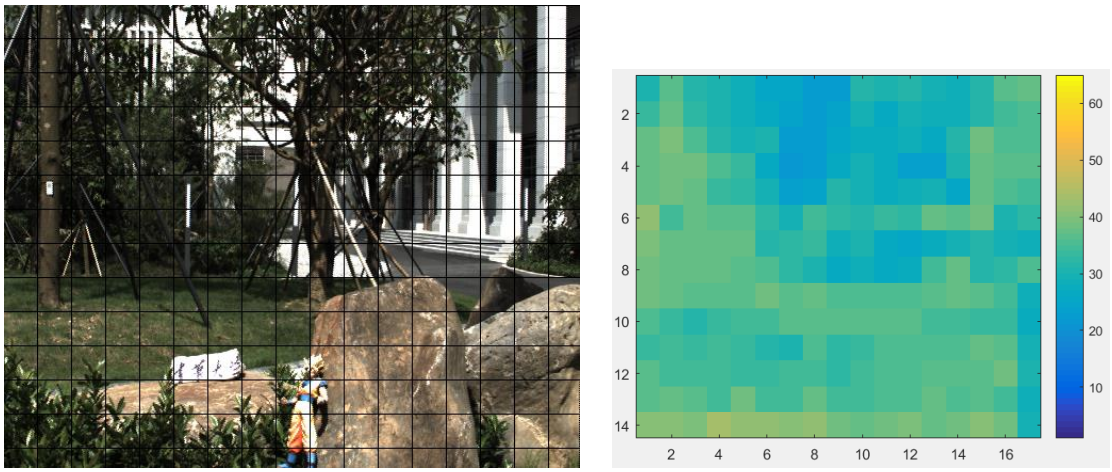
NagoyaFujita_blocks_num=812



NagoyaOrigami_blocks_num=812



Toys_blocks_num=238



Trees_blocks_num=238

四、 结论与思考

这次《凸优化》课程大作业的模型方面，我们具有以下的三个创新点：

首先，我们创新性地使用了以权重为优化变量的思想，而不是将待预测图片作为优化的变量（以待预测图片作为优化变量的方法，在课程的小上机作业中被大部分同学所使用），同时目标优化函数采用的是前一帧图像的 R5 的真实值和预测值之间的范数损失。

除此之外，我们也将前一帧图像中的不同位置的图片的先验知识进行归纳，并根据前后两帧同一位置的 block 的具体特征进行权重形式分类讨论。如果前后两帧同一位置的 block 的 PSNR 很大，即采用矩阵形式的权重，进而突出图片的细节特征；如果前后两帧同一位置的 block 的 PSNR 较小，即图片在前后两帧中的动态特征较为明显，则采用单个数值形式的权重，从而进行分块级别的优化，防止使用权重矩阵时出现图像的过拟合。

最后，我们在选择合适的权重矩阵时，对于阈值的考虑也是比较科学的，没有通过手动尝试的方法，我们是将所有 block 对应的 PSNR 进行了统计，根据每张图片统计出来的 PSNR 直方图的“峰值”和“谷值”进行对应阈值的选取。通过实际的测试，在七张图片平均 PSNR 较高的情况下，对于每一个图片得到了最好的 PSNR 的数值结果。

对于模型我们也有进一步的优化思路：

首先，可以尝试在损失函数的计算中加入正则化项，进而增加模型的鲁棒性，这个方案我们尝试过，不过由于计算复杂度太高导致计算过程极为缓慢，在模型指标和运算时间代价的权衡下，我们放弃了加入正则项这个方法。

除此之外，我们可以借鉴其他方法所得到的结果与我们现在所得到的结果再进行模型融合，从而得到更优的模型结果。例如我们在小上机作业中所采用的以待预测图片作为优化变量的这一思路得到的模型，其结果与我们的模型得到的结果进行加权取平均等融合操作，也许会得到更好的结果。

这次实验我们使用凸优化的 CVX 包，完成了“多视图视频编码的帧间预测”，通过分析问题，并将预测问题归纳为一个凸优化问题，寻找优化目标函数以及约束条件，完成了图片的预测，在 PSNR 的模型评价指标下，我们的算法实现了较

好的预测效果。本次实验为我们将来的课题研究积累了实践的经验，也提高了我们将普通问题转化为凸优化问题来思考并解决的能力。

五、 附录

5.1 报告中所有参数的具体含义

$Rx(i,j)$：在图x中水平方向的第i个、在垂直方向的第j个 block；

$block_{size}$：block 的边长（这里取值范围在 4.8.16.32.64 中）；

$block_{num\,row}$：在水平方向 block 的总个数；

$block_{num\,col}$：在竖直方向 block 的总个数；

$Offset$：计算用于优化问题的$Rx(i,j)$是，偏移量的范围；

$\omega_x(i,j)$：位置在i，j的 block 对应的权重矩阵；

$\omega_{xbest}(i,j)$：经过计算得到的，位置在i，j的 block 对应的最优的权重矩阵；

$\widehat{R5}$：图像 R5 的预测值；

$\hat{X}$：图像 X 的预测值；

$PSNR\left(X_R,\widehat{X_R}\right)$：在 R 通道中的 PSNR 的数值；

$PSNR_{mean}$:RGB 三通道的 PSNR 的平均值。

5.2 代码的使用说明

1、运行程序前需将 7 组图片添加到 img 文件夹中该图片对应的文件夹内；

2、运行 running.m 文件，开始预测所有 7 张图片并输出结果（running.m 文件会循环调用 main.m 文件）；

3、运行 show_psnr_means.m 文件，输出所有 7 张图片，前后两帧上下左右图片分块的 psnr 的频率直方图；

4、运行 show_block.m 文件，保存所有 7 张图片的分块结果；

5、运行 show_psnr_blks.m 文件，输出所有 7 张图中，每个小块的 psnr 热力图。

输出文件的具体信息如下（这里以运行 Boys 图片组为例）：

img/Boys/X.png 为原图；

img/Boys/X_final.png 为最佳的预测结果；

img/Boys/X_blocks.png 是对 X_final.png 的分块情况；

img/Boys_psnr_blks.mat 保存了预测结果图片中每块的 psnr 指标；

img/Boys_psnr_final.mat 保存了预测结果图片总体的 psnr 指标；

img/Boys_psnr_means.mat 保存了上下帧对应的每块图片之间的差别指标，其中 psnr_means 越大，表示该块图片上下帧之间差别越小；

5.3 组内分工

组员	具体分工
谭世琦	1.基于权重矩阵的优化问题的代码实现，以及可视化代码的实现（PSNR 热力图等）； 2.报告结果以及实验过程部分撰写； 3.PPT 制作。
杨毅远	1.基于数值型权重的优化问题的代码实现，以及可视化代码的实现（PSNR 直方图等）； 2.报告总结以及实验过程部分撰写； 3.PPT 制作。
共同完成	前期调研以及两模型融合的代码实现。