

# BM3D 去噪及伪彩色变换

\* Name: SizheWei(魏思哲) sizhewei@sjtu.edu.cn

## 目录

<b>1 实验原理</b>	<b>2</b>
1.1 BM3D 去噪实验原理 . . . . .	2
1.1.1 第一步：基础估计 . . . . .	2
1.1.2 第二步：最终估计 . . . . .	3
1.2 伪彩色变换实验原理 . . . . .	4
<b>2 算法思路</b>	<b>4</b>
2.1 BM3D 去噪设计思路 . . . . .	4
2.2 伪彩色变换设计思路 . . . . .	7
<b>3 实验结果</b>	<b>7</b>
3.1 BM3D 实验 . . . . .	7
3.1.1 相同参数，比较不同分辨率的图像经过 BM3D 变换所需时间 . . . . .	7
3.1.2 相同图片，比较阈值不同带来的影响。 . . . . .	8
3.1.3 不同阈值对相同图片处理效果对比 . . . . .	9
3.2 伪彩色变换实验 . . . . .	10
<b>4 思考拓展</b>	<b>11</b>
4.1 实验思路 . . . . .	11
4.2 实验过程 . . . . .	11

# 1 实验原理

## 1.1 BM3D 去噪实验原理

BM3D 算法是降噪算法中最突出的算法之一。在本文的 1~4 节，我们将率先考虑针对与灰度图的 BM3D 去噪算法，对于彩色图像的处理将在本文第 5 节进行叙述。另外，本文着重考虑图片中的噪声为高斯白噪声，方差为  $\sigma^2$ 。BM3D 去噪算法主要分为两个步骤 [1]：

Step 1 第 1 步：基础估计 [2]

- 块估计，对每一个图像块进行如下操作：
  - 分组：将与正要处理的图像块相似的图像块并将他们组合成一个三维矩阵；
  - 阈值收缩协同过滤：将三维矩阵的第三维进行变换，变换后接近零的系数置零，逆变换之后将图像块放回到原来的位置；
- 综合：由于有一些图像块的重叠，用非零元素的个数作为权重，以减轻混叠的影响；

Step 2 第 2 步：最终估计

- 块估计，对每一个图像块进行如下操作：
  - 分组：对每一个图像块找到其第一步基础估计中对应图像块的相似图像块三维矩阵以及原噪声图中的对应的图像块的相似图像块三维矩阵，共两个；
  - 维纳滤波-协同过滤：对上述两个矩阵分别进行第三维上的小波变换，用基础估计得到图像块的小波变换作为滤波器，对噪声图进行滤波，将得到的图进行反变换，变换后的图像块将每个图像块放回原来的位置；
- 综合：同第一步中的综合方式，对图像块进行一个加权处理，从而减轻混叠的影响。

这两个步骤的原理我们将在本小节中进行阐述。Figure 1 展示了 BM3D 算法的整体流程。

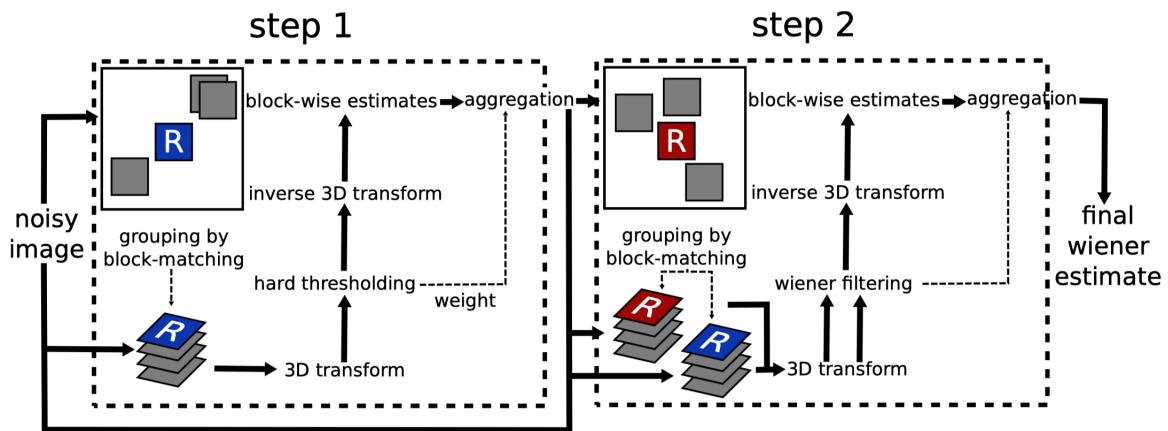


Figure 1: BM3D 去噪算法流程图 [2]

### 1.1.1 第一步：基础估计

**分组：**对于每个目标图块，都在其附近的邻域（可以设定该邻域大小）中寻找相似度较高的图块归为一组。相似度可以近似看作与欧式距离成反比。将这些图块整合成一个三维矩阵。利用等式 (1) 中给出的条件来判断每个块是否符合相似的要求。

$$\mathcal{P}(P) = \{Q : d(P, Q) \leq \tau^{hard}\} \quad (1)$$

在等式 (1) 中：

- $\tau^{hard}$  为距离的硬阈值，若距离  $d < \tau^{hard}$ ，则认为两个图像块相似；
- $d()$  为计算图像块  $P, Q$  之间距离的函数，有很多计算图像块之间距离的方法，本文中将采用  $\ell^p - norm(p = 2)$ ，即欧式距离进行计算；
- 在这其中还需要额外注意的是，在比较两个图片块相似程度前需要对图片块进行 DCT 变换（或其他小波变换），以降低噪点的干扰。

**协同过滤：**将分组得到的三维矩阵进行第 3 个维度的硬阈值收缩，再将第三维中非零的个数计算出来。见等式 (2)。

$$\begin{aligned} \mathbb{P}(P)^{hard} &= \tau_{3D}^{hard^{-1}}(\gamma(\tau_{3D}^{hard}(\mathbb{P}(P)))) \\ \gamma(x) &= \begin{cases} 0 & \text{if } |x| \leq \lambda_{3D}^{hard}\sigma \\ x & \text{otherwise} \end{cases} \end{aligned} \quad (2)$$

其中：

- $hard$  作为上角标表示进行硬阈值收缩操作或经过了硬阈值收缩之后的变量；
- $\gamma$  为一个硬阈值操作的操作算子，其阈值为  $\lambda_{3D}^{hard}\sigma$ ；
- $\tau_{3D}^{hard}$  表示采用的变换方式，如 DCT 变换，其中  $3D$  表示对第三维进行该操作， $\tau_{3D}^{hard^{-1}}$  为其反变换。

**综合处理：**将经过三维滤波的图片块，经过协同过滤中得到的非零元素个数做权重叠加，从而得到初步的图片块滤波估计。

对于每个像素点都有一个对应的基础估计，可以表示为等式 (3)

$$\forall Q \in \mathcal{P}(P), \forall x \in Q, \begin{cases} \nu(x) = \nu(x) + \omega_P^{hard} \mu_{Q,P}^{hard}(x) \\ \delta(x) = \delta(x) + \omega_P^{hard} \end{cases} \quad (3)$$

在等式 (3) 中，有：

- $\nu$  (或  $\delta$ ) 表示在第一步估计中得到每个点的处理后的值的分子 (或分母)；
- $\mu_{Q,P}^{hard}(x)$  是图像块  $Q$  根据参考块  $P$  得到的每个点的参考值；
- $\omega_P^{hard} = \begin{cases} (N_P^{hard})^{-1} & \text{if } N_P^{hard} \geq 1 \\ 1 & \text{otherwise} \end{cases}$ ；
- $N_P^{hard}$  是经过第三维变换之后得到的非零元素的个数；

$$\mu_{Q,P}^{basic}(x) = \frac{\sum_P \omega_P^{hard} \sum_{Q \in \mathcal{P}(P)} \mathcal{X}_Q(x) \mu_{Q,P}^{hard}(x)}{\sum_P \omega_P^{hard} \sum_{Q \in \mathcal{P}(P)} \mathcal{X}_Q(x)} \quad (4)$$

### 1.1.2 第二步：最终估计

**分组：**与第一步中的分组操作相似

$$\mathcal{P}(P) = \{Q : d(P, Q) \leq \tau^{wie}\} \quad (5)$$

**协同过滤：**进行维纳滤波的系数  $\omega$  见等式 (6)。

$$\omega_P(\xi) = \frac{|\tau_{3D}^{wien}(\mathbb{P}^{basic}(P))(\xi)|^2}{|\tau_{3D}^{wien}(\mathbb{P}^{basic}(P))(\xi)|^2 + \sigma^2} \quad (6)$$

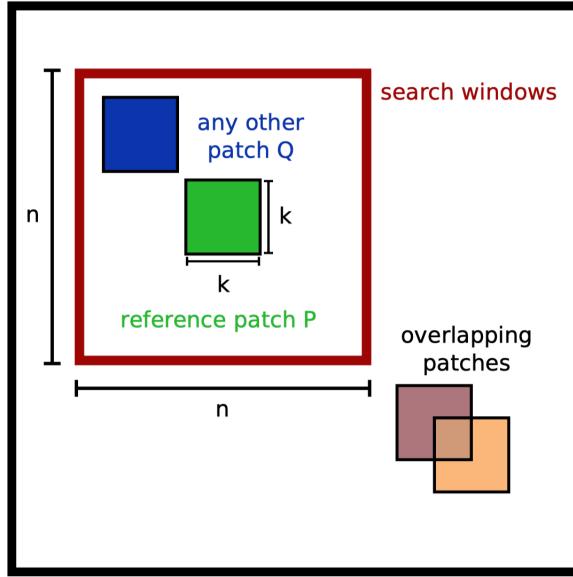


Figure 2: 图像块，搜索范围和混叠示意图

硬阈值收缩操作见等式 (7)。

$$\mathbb{P}^{wien}(P) = \tau_{3D}^{wien-1}(\omega_P \cdot \tau_{3D}^{wien} \mathbb{P}(P)) \quad (7)$$

综合处理：将经过三维滤波的图片块，通过等式 (8) 对每个像素点做最终的估计。

$$\forall Q \in \mathcal{P}(P), \forall x \in Q, \begin{cases} \nu(x) = \nu(x) + \omega_P^{wien} \mu_{Q,P}^{wien}(x) \\ \delta(x) = \delta(x) + \omega_P^{wien} \end{cases} \quad (8)$$

在等式 (9) 中，有：

- $\omega_P^{wien} = \|\omega_P\|_2^{-2}$ .

为了减轻边缘效应，最后添加一个 Kaiser 窗，见等式 (9)。

$$\mu^{final}(x) = \frac{\sum_P \omega_P^{wien} \sum_{Q \in \mathcal{P}(P)} \mathcal{X}_Q(x) \mu_{Q,P}^{wien}(x)}{\sum_P \omega_P^{wien} \sum_{Q \in \mathcal{P}(P)} \mathcal{X}_Q(x)} \quad (9)$$

## 1.2 伪彩色变换实验原理

由于人眼对灰度微小的变化并不敏感，而对彩色细小变化极为敏感，因此，我们利用这一点，通过将灰度变化转换为色彩变化，来实现对图像信息的增强，这便是伪彩色变换的基本原理。

## 2 算法思路

### 2.1 BM3D 去噪设计思路

第一步与第二步过程中有很大一部分重叠，因此在算法思路中，第二步中的过程省略，可参考第一步中对应部分的算法思路。Algorithm 1 为第一步基础估计的算法设计，第二步与此类似。

对应代码中函数: *BM3D\_step1()* 和 *BM3D\_step2()*。

---

**Algorithm 1:** 第一步: 基础估计

**Input:** 噪声图

**Output:** 经过基础估计得到的图片

**Result:** 返回经过基础估计之后的到的图像块:

```
1 初始化参数, init() 得到空图像、空权重表、Kaiser 窗;
2 for 每一行的步长 do
3   for 每一列的步长 do
4     设定一个当前块的坐标值;
5     块匹配得到相似块集合、位置、数量;
6     协同过滤吼得到的饿相似块集合, 以及非零项数量;
7     综合处理, 非零项数量作为权重加权处理, 处理后返回远处。
8   end
9 end
```

---

对于每一步中, 找到当前的坐标值, 这里将每个图像块的左上角的定点作为了参考点, 因为参考点即使在整个图像内, 但是这个图像块中其他元素可能会超出图像范围, 因此我们做一个判断, 倘若参考点对应的图像块中有元素超出图像范围则将图像块的参考点设定为恰好不超范围的临界点。

---

**Algorithm 2:** 确定当前块坐标值

**Input:** 当前位置 i,j; 步长; 图像块大小; 图像宽度、高度

**Output:** 参考图像块顶点 x,y

```
1 if 位置 i× 步长 + 图像块大小 < 图像宽度 then
2   | x = 位置 i× 步长
3 else
4   | x = 图像宽度-图像块大小
5 end
6 if 位置 j× 步长 + 图像块大小 < 图像高度 then
7   | y = 位置 j× 步长
8 else
9   | y = 图像高度-图像块大小
10 end
```

---

根据当前坐标值, 在一个合适的搜索范围内对每个块进行处理, 找到最相似的几个图像块放入到三维矩阵中。见 Algorithm 3。

---

### Algorithm 3: 分组

---

**Input:** 噪声图; 当前块坐标值  
**Output:** 相似块三维矩阵; 相似块坐标; 相似块数量

```
1 初始化一些数据结构用于记录结果;
2 设定搜索图像块的范围 for 每行范围 do
3   for 每列范围 do
4     对应图像块进行 DCT 变换;
5     计算与参考图像块的距离;
6     if 距离 < 阈值 then
7       将图像块放入三维矩阵中;
8       相似块数量 +=1;
9       将图像坐标加入二维矩阵中;
10      将图像块与参考图像块之间的距离加入到一位矩阵中;
11    end
12  end
13 end
14 将统计好的三维矩阵按照每一个图像块距离从小到大排序;
15 if 相似块数量 > 最大要求数量 then
16   | 只取三维矩阵中前'最大要求数量'个图像块;
17 end
```

协同过滤目的是将三维矩阵中的第三维进行小波变换，并进行阈值收缩，为了方便后面的综合处理，这里将阈值收缩之后非零元素的数量进行统计。见 Algorithm 4。

---

### Algorithm 4: 3D 协同过滤

---

**Input:** 相似图像块三维矩阵  
**Output:** 相似块集合; 非零项数量  
**Result:** 返回经过基础估计之后的到的图像块:

```
1 初始化参数, init() 得到空图像、空权重表、Kaiser 窗;
2 for 每一图像块的宽 do
3   for 每一图像块的高 do
4     将对应的第三维取出来进行 DCT 变换;
5     变换后与阈值比较, 小于阈值的置零并统计非零项数量;
6     反变换,
7   end
8 end
```

最后就是综合处理算法，将协同过滤中的非零元素数量作为权重放置到空权重表中，将每个图像块进行 IDCT 变换后放置回原来的位置。见 Algorithm 5。

---

**Algorithm 5:** 综合处理

---

**Input:** 相似块三维矩阵; 相似块坐标; 空图像; 空权重表; 非零项数量; 相似块数量;  
Kaiser 窗

**Result:** 返回经过基础估计之后的到的图像块:

```
1 初始化参数, init() 得到空图像、空权重表、Kaiser 窗;
2 if 非零项数量 < 1 then
3   权重 = 1
4 end
5 for i= 相似块数量 do
6   每个图像块进行 IDCT;
7   反变换后的图像块放置回空图像中对应位置;
8   每个图像的权重放置回空权重表的对应位置;
9 end
```

---

## 2.2 伪彩色变换设计思路

伪彩色变换有如下两种设计思路:

- 1) 伪彩色增强, 通过灰度与颜色的对应关系, 查找图片中每个像素点的灰度值, 进而查找其颜色值, 用相应的彩色替代;
- 2) 伪彩色变换, 通过对 RGB 三通道的值进行一定的变换, 组成该像素点的彩色值。

## 3 实验结果

### 3.1 BM3D 实验

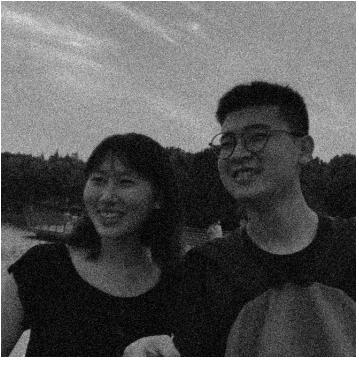
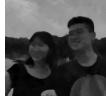
在进行 BM3D 去噪实验中, 我设计了 4.1.1~4.1.3 所示的 3 个实验, 从以下几个角度观察和测试 BM3D 去噪的能力:

- 1) 图片分辨率大小对 BM3D 去噪的时间影响 (4.1.1);
- 2) 不同阈值对相同图片处理的速度和效果 (4.1.2);
- 3) 不同图片类型 (人物、风景、动物) 去噪效果 (4.1.3)。

#### 3.1.1 相同参数, 比较不同分辨率的图像经过 BM3D 变换所需时间

在相同的参数下, 我们选择了一组图片纹理完全相同, 仅有像素点不同的一组图片经过 BM3D 变换测试时间, 如 Table 1. 可见随着图片像素点增多, BM3D 去噪时间增大得很快, 耗时较长。

Table 1: 不同分辨率下相同参数进行 BM3D 去噪对比

	<b>192×192</b>	<b>428×428</b>	<b>664×664</b>
<i>Origin</i>			
<i>Noise</i>			
<i>Basic</i>			
<i>Final</i>			
<i>Runtime(s)</i>	414.76	2246.19	5400.41

### 3.1.2 相同图片，比较阈值不同带来的影响。

我们选取同样的一张图片，由于像素块大小、以及图像大小尺寸不同，我们按照这个比例来设计阈值，见等式 (10)：

$$\lambda_{3D}^{hard} \sigma = \mathcal{K} \times n^2 \quad (10)$$

其中：

- $\mathcal{K}$  是我们定义的一个比例系数，通过更改其值来改变阈值
- $n$  是图像块宽度

我通过选取  $\mathcal{K}$  分别为 75, 125(as default), 175 来观察去噪效果，见 Table 2。

Table 2: 不同分辨率下相同参数进行 BM3D 去噪对比

	<b>75</b>	<b>125</b>	<b>175</b>	
<i>Origin</i>				
<i>Noise</i>				
<i>Basic</i>				
<i>Final</i>				
<i>Runtime(s)</i>	402.89	398.22	396.80	
<i>PSNR</i>	28.042	28.236	28.42	

### 3.1.3 不同阈值对相同图片处理效果对比

考虑到图片的纹理可能会对去噪效果产生影响，因此我们选择了四组图片，分别为人像，动物（猫），景色（白天），景色（夜间）进行去噪测试，见 Table (3)：

Table 3: 不同类型图片去噪效果对比

	<i>people</i>	<i>animal</i>	<i>scape-day</i>	<i>scape-night</i>
<i>Origin</i>				
<i>Noise</i>				
<i>Basic</i>				
<i>Final</i>				
<i>Runtime(s)</i>	414.76	398.22	429.92	425.58
<i>PSNR</i>	30.885	28.236	27.143	29.62

### 3.2 伪彩色变换实验

我们选择了三组图片进行两种模式的伪彩色变换，见 Figure 3:

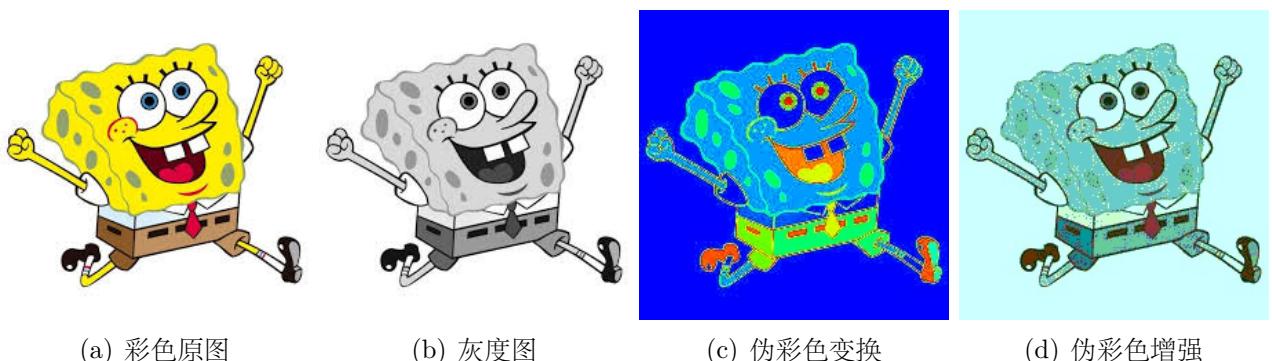


Figure 3: 伪彩色图片展示 (1)

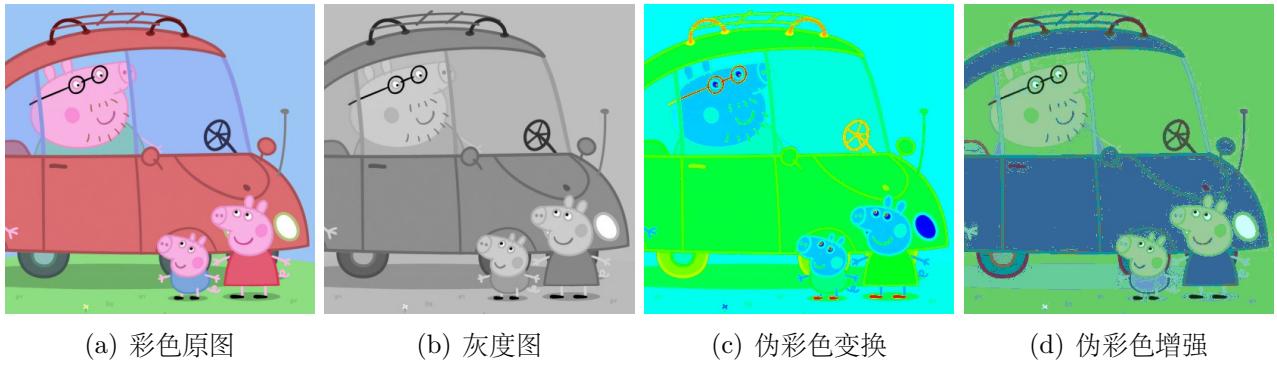


Figure 4: 伪彩色图片展示 (2)

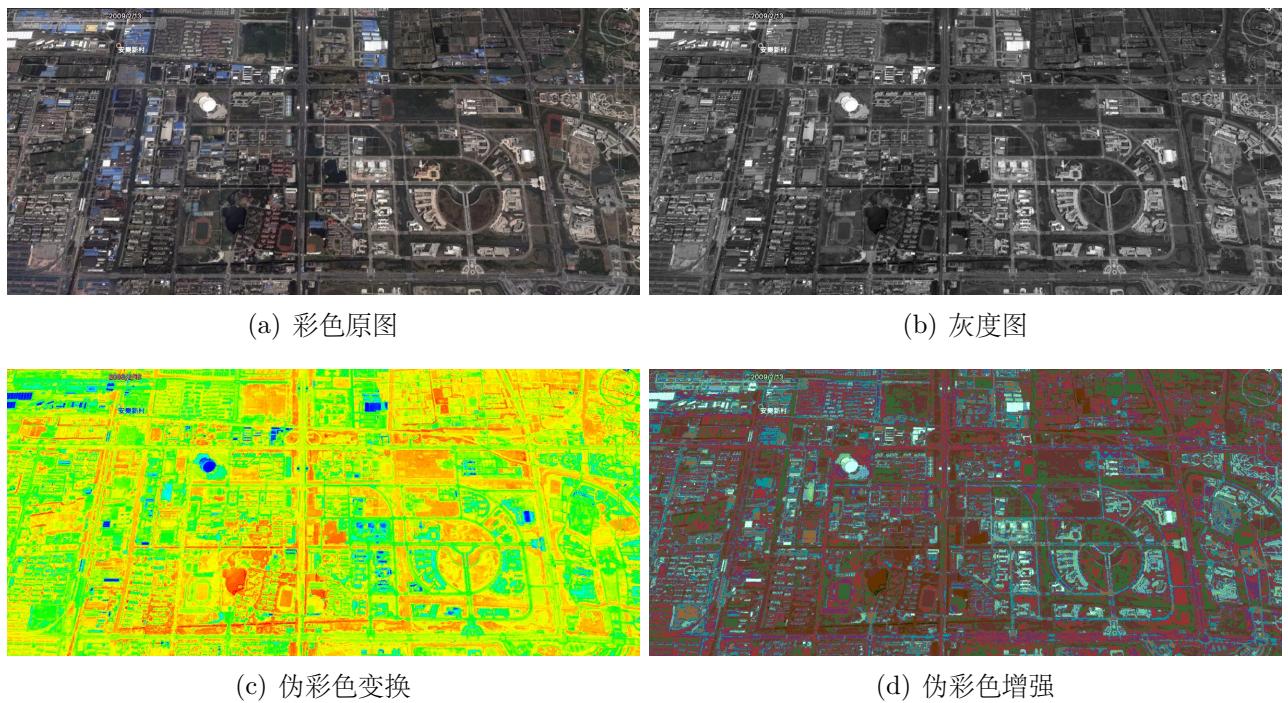


Figure 5: 伪彩色图片展示 (3)

## 4 思考拓展

### 4.1 实验思路

在有关灰度图像部分考虑完成之后，根据老师的指导，我开始思考彩色图像的 BM3D 去噪过程，由于彩色图像可以在 YUV 空间域进行处理，我决定将图像 Y 通道进行分组块匹配，同时按照相同的位置对 U、V 通道进行同样的分组，得到的三个三维矩阵分别进行协同滤波和综合处理，最后将三通道合并到一起，从而得到去噪后的图像。[\[2\]](#)

### 4.2 实验过程

在灰度图的处理上，我将代码进行一系列的修改，但是得到的结果不尽人意。见 Figure 4。调试代码的过程中，我认为错误应该出现在数据类型的处理上，在我进行分组之前，通过 `merge()` 函数，我将 YUV 通道分开，但是分别进行 3D 协同滤波之后，U、V 通道的数据不再是 `uint8` 类型 (`[0, 255]`)，然后我便将其通过 `int8()` 函数处理，转换到对应的数据类型，再进行反变换，但是

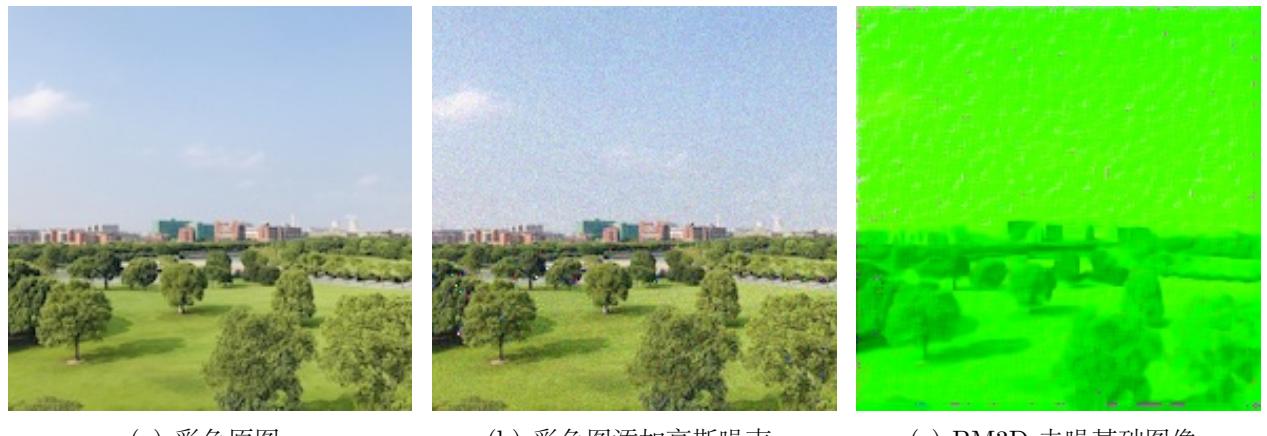


Figure 6: 彩色图片 BM3D 去噪尝试失败示意图

具体原因还需进一步实验。这一部分我将在课程之余向老师与学长请教，并与同学探讨错误原因，再解决后补充进来。

## References

- [1] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising with block-matching and 3d filtering. In *Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning*, volume 6064, page 606414. International Society for Optics and Photonics, 2006.
  - [2] Marc Lebrun. An analysis and implementation of the bm3d image denoising method. *Image Processing On Line*, 2:175–213, 2012.