对马赛克瓷砖选取与优化方案的研究与设计

摘要

本文采用RGB三维数据量化表示图像颜色，并借助欧氏距离衡量颜色之间的相关程度；借助欧氏距离的结果并利用遗传算法寻求最优解；使用蒙特卡罗法模拟计算色彩色覆盖率，从而求解三个问题。

第一问，对给出的图像颜色求其与已有瓷砖颜色RGB值欧氏距离的最小值以寻求最符合的瓷砖颜色。由于在求解的过程中存在单一变量差值较大的情况，本文考虑到颜色误差值，即附件颜色与瓷砖颜色的RGB值作差值并求方差，但在多次实验后我们发现其对该问

题的影响较小，故忽略该变量产生的误差。

第二问结合第一问欧氏距离的计算思路选取最能提高图像表现度的颜色。我们采用使新添加颜色与已有颜色欧氏距离最小值作为遗传算法的适应度函数，使函数值达到最大的变量值即为所求。为保证新添加的颜色在最大程度上满足题目要求，本文采用了贪心算法在最大程度上提高颜色对图像拼接作用的效果。

第三问结合第二问的遗传算法，考虑瓷砖成本和颜色覆盖度的综合影响。首先对瓷砖成本做正向化处理，成本量化转为极大值指标；其次通过蒙特卡罗法估计已有的瓷砖颜色和新添加颜色对于整体色彩空间的覆盖率；标准化两个指标后对其进行评分；最后通过适应度函数对研发成本和覆盖率进行加权计算出其综合影响效果，在综合效果中获取最优。

最后本文还对实现颜色应用的具体方案给出了建议，对各模型在实际中的应用价值进行了详细的讨论，并提出了解决方案。

关键字：遗传算法 贪心算法 欧氏距离 TOPSIS 优劣解距离法

一、 问题重述

边长不超过 5cm 的马赛克瓷砖可以依据用户需求拼接成文字或图案，但由于瓷砖颜色有限，在拼接时只能根据原图颜色选择瓷砖，故在呈现照片时会有些许偏差。

问题一是基于附件 2 附件 3 提供的图像 1 和图像 2 的颜色选择与其相近的瓷砖颜色，并将选择结果输出至结果文件 result1.txt 和 result2.txt 中。

问题二考虑到现有的马赛克瓷砖的局限性，为增强图像表现力和准确度，不考虑研发难度，如若增加十种颜色的瓷砖应增加何种颜色。

问题三则是同时考虑到成本和表现效果，应新增加何种数量何种颜色的马赛克瓷砖种类。

二、 问题分析

2.1 问题一的分析

图像的颜色由 R、G、B 三个变量共同决定，所以在问题一中通过将附件中颜色的RGB值与附件一中的RGB值作差的平方和求其最小值，可以得到最接近的颜色；但考虑到两个颜色会有某一值偏差较大的情况存在，我们对附件颜色与瓷砖颜色的RGB值作差值并求方差可得到误差值。通过对平方和及误差值的加权，可得到与附件颜色最接近的瓷砖。

2.2 问题二的分析

为提高拼接图像的表现力，我们借助问题一的近似性度量函数来选择需要新添加的颜色，我们选择颜色的标准是保证新添加的颜色与已有的 22 种颜色近似性相差最大，使加入后的颜色分布均匀，在供给侧层面提高拼接图像的表现力。由于RGB值来标准化颜色共有$2^8 * 2^8 * 2^8 = 16777216$种颜色，并且每一个颜色的RGB值对应一个三维空间坐标，易将其转化为二进制坐标，故考虑通过遗传算法来寻找最优解。同时采用贪心算法来添加新的颜色使其均匀分布来提高拼接图像的表现力。

2.3 问题三的分析

当综合考虑成本和表现效果时，关于表现效果的评价，依据现有瓷砖颜色对于整个颜色

空间的覆盖率，即每一个瓷砖颜色存在一个近似颜色范围，已有颜色总的近似颜色范围与整个颜色空间的比值为覆盖率。本文通过蒙特卡罗模拟来计算覆盖率。成本的评价考虑，通过已选颜色瓷砖的个数衡量成本大小。综合考虑时借助优劣解距离法思想标准化两个评价量，并增加权重系数来衡量综合效果。最后对其做灵敏度分析，改变权重来分析选择结果。

三、 模型假设

1、假设原始图像为 24 位真彩色格式;
2、假设用户图像的色彩 R、G、B 三个变量的分布均匀;
3、RGB颜色空间分布均匀。

四、 符号说明

符号	说明
X_{ik}	附件一中瓷砖颜色的RGB值
X_{jk}	附件二和附件三中颜色的RGB值
t	RGB预测值的差值
d	两点之间的欧氏距离
w_i	变量在评价过程中所占有的比重
r	颜色在空间中的适用半径

五、 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 近似值度量模型的建立

问题一所要解决的问题包括使用 MATLAB 提取附件的RGB数据以及在现有瓷砖的颜色中获取最接近附件图像颜色的瓷砖编号。

为获取最接近附件图像的瓷砖颜色，我们将RGB三维数据进行向量坐标处理，即得到用户图像RGB对应的三维向量$X_i=(x_{i1},x_{i2},x_{i3})^T$和瓷砖RGB对应的三维向量$X_j=(x_{j1},x_{j2},x_{j3})^T$，这两个向量之间的距离可以被用作聚类分析的数量指标，从而可定量进行聚类分析。

对于两个 RGB 三维向量的相似性度量以最小欧式距离函数来衡量进行分析[1]：即求每一个图像颜色的 RGB 坐标与瓷砖颜色中二十二个RGB坐标之间的距离，距离最小值所对应的瓷砖编号即为所求。公式如下:

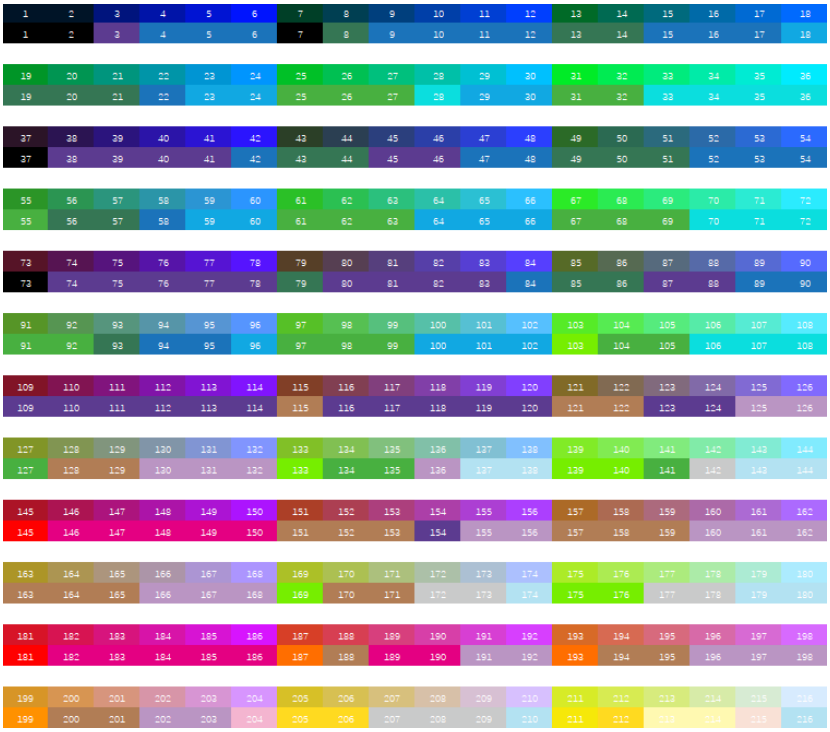
$$\min l = min\sqrt{\left(X_i - X_j\right)^2}$$

$$= min\sqrt{\left(\left(x_{i1}-x_{j1}\right)^2+\left(x_{i2}-x_{j2}\right)^2+\left(x_{i3}-x_{j3}\right)^2\right)} \quad i\in[1,22],j\in[1,216]\text{（附件二）}$$

$$i\in[1,22],j\in[1,200]\text{（附件三）}$$

5.1.2 模型求解

求得结果如图所示，其中单数行色号为原始图像颜色，双数行色号为求解所对应瓷砖的颜色：



5.1.3 模型的改进

可以看到原始 RGB 数据的颜色与通过最小距离算法求得的瓷砖颜色有些许偏差，故提出偏差概念，即判断是否存在 RGB 中某一变量偏差较大的情况。偏差即为 RGB 三个变量差值的方差[2]，计算公式如下：

$$m_k = x_{ik} - x_{jk}$$

$$\bar{m} = \frac{1}{3}\sum_{k=1}^{3} m_k$$

$$t = \sqrt{\sum_{k=1}^{3}(m_k - \bar{m})^2}$$

求得 t 即为偏差值，对最小距离算法求得的结果进行偏差分析，并进行进一步调整，分析与所选颜色相近的前三种颜色结果如下图：

				RGB
Dist	Std	Nnumber	Color7	0,63,39
5490	31.7962	1		0,0,0
7859	5.2915	7		53,118,84
18578	44.0341	5		72,176,64

				RGB
Dist	Std	Nnumber	Color21	0,149,125
5451	51.6269	7		53,118,84
5606	48.1352	6		27,115,186
9634	67.6486	5		72,176,64

				RGB
Dist	Std	Nnumber	Color86	43,106,125
1237	23.6291	7		53,118,84
5420	49.6924	5		72,176,64
6089	54.5069	11		92,59,144

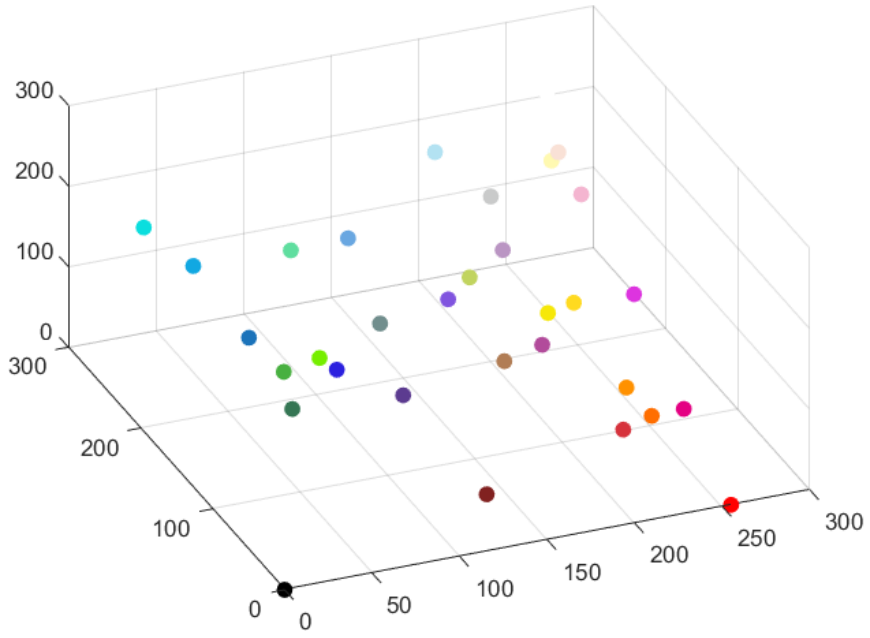
				RGB
Dist	Std	Nnumber	Ccolor135	129,192,125
7226	24.9065	5		72,176,64
8393	60.136	10		117,125,85
9998	61.8304	22		186,149,195

分析图像一中的四种颜色，发现与提供的颜色的RGB值距离最近的颜色，都可以在人工检验下通过，所以偏差对于所求得颜色的影响较小基本可以忽略，故偏差对数据的影响忽略不计。

5. 2 问题二模型的建立与求解

为获取需要增加的颜色，我们从实际需求和色彩均匀分布角度出发，保证添加的颜色与瓷砖已有的 22 种颜色差距最大，使添加后瓷砖颜色的RGB值在三个维度分布均匀。我们依据确定的原则，采用遗传算法预测能使拼接效果达到最优的颜色组合。

我们利用 MATLAB 的 scatter3 函数表现已有的瓷砖颜色在三维空间中的散点分布，具体分布如图展示:



5. 2. 1 遗传算法的建立

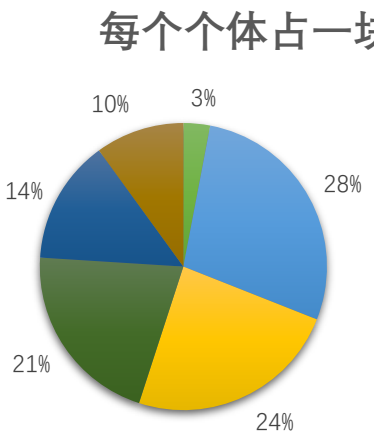
基于遗传算法[3]的求解步骤如下:

$Step1$: 初始化。对遗传算法的控制参数进行初始化，通过随机数生成函数随机生成了100 条染色体，同时设置最大迭代次数Gen =15000、最大交叉概率P_c =0.5、最大变异概率P_m =0.2;

$Step2$: 计算适应度函数值。将每一条染色体放在适应度函数中，得到适应度结果;

$Step3$: 算法的终止判断。判断是否达到最大迭代次数，若达到则输出最优解；否则则进入下一步;

$Step4$: 通过轮盘赌法进行自然选择。计算每个个体适应度占总适应度的比例，总适应度是一个饼图，每个个体占据一定的扇形区域。之后通过不断生成 0-1 的随机数对个体进行自然选择，而适应度高的个体更容易被选择;



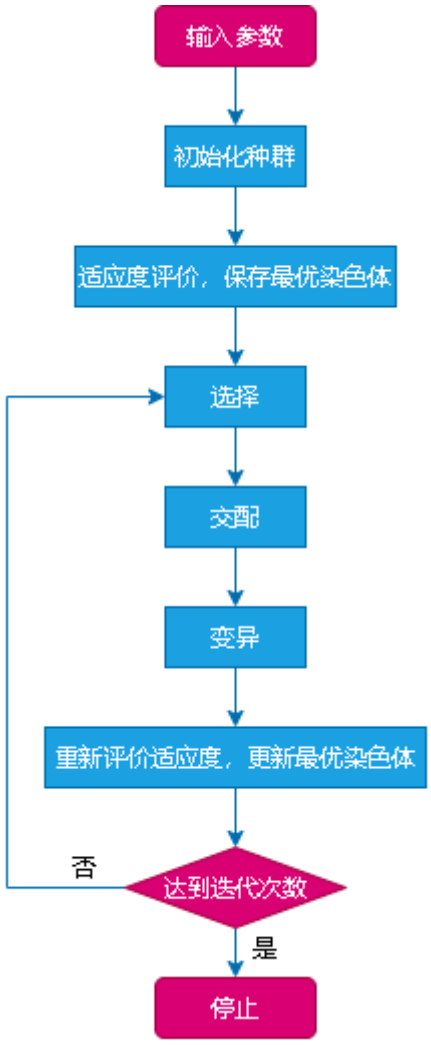
$Step5$: 交叉运算。每两条染色体组合在一起，将两者的部分染色体进行交换，而我们

采用的单点交换即为选择染色体上的一个基因点，从这个基因点开始的两条染色体片段进行互换；

Step6：变异运算。我们采用基本位变异，即选择一条染色体的一个基因点，将其取反，即将“0”变为“1”，或将“1”变为“0”；

$Step7$：获得最优解。迭代结束之后，我们求得使适应度函数最大即适应度最大的染色体，就是我们需要的能使拼接效果达到最优的瓷砖颜色。

遗传算法求解流程如下：



我们在数据的计算中应用贪心算法，即：在每一次迭代后依据已有的条件选取最优。在求得最优解之后，我们将代表需要新增瓷砖颜色 RGB 值的二进制编码通过简单的函数进行解码，获得十进制的数值。

5. 2. 2 选择适应度函数

其中$Step2$中适应度函数，我们依据目标函数的值，将个体的表现通过数值评价表现出来，我们的适应度函数在构建时保证添加颜色与已有颜色相差最大的原则进行优化。适应度函数公式如下：

定义$d(A_0, A_i)$为添加颜色与瓷砖已有颜色 RGB 值在三维坐标中的距离，即：

$$d(A_0, A_i) = \sqrt{(x_0 - x_i)^2 + (y_0 - y_i)^2 + (z_0 - z_i)^2}$$

并且说明m为已有瓷砖的颜色个数，$m \geq 22$

首先我们定义的适应度函数为添加颜色与已有点欧氏距离和的倒数；

$$f_1(x_0, y_0, z_0) = \frac{1}{\sum_{i=1}^{m} d(A_0, A_i)}$$

我们在运行过程中取f_1的最小值，其值越小说明添加颜色与已有颜色 RGB 值的欧氏距离越大，即添加颜色与已有颜色相差越大越符合我们的要求。但在实际导出的过程中，我们发现大部分结果与已有瓷砖颜色的差距较小，甚至存在颜色相重合的情况，故我们进行适应度函数的优化，即定义适应度函数为添加颜色与已有点欧氏距离乘积的倒数，来避免两个颜色色差过小，排除过于相似的颜色；

$$f_2(x_0, y_0, z_0) = \frac{1}{\prod_{i=1}^{m} d(A_0, A_i)}$$

我们在运行过程中仍取f_2的最小值，其值越小说明添加颜色越符合我们的要求。但在迭代过程中效果仍不显著，存在导出颜色相似的现象，故我们再度进行适应度函数的优化，即定义适应度函数为添加颜色与已有点欧氏距离最小值；

$$f_3(x_0, y_0, z_0) = min\big(d(A_0, A_i)\big)$$

我们在运行过程中取f_3的最大值，f_3的值越大越符合我们的期望，即添加颜色与已有颜色相差越大。

5. 2. 3 遗传算法模型求解

我们通过 15000 次遗传算法的迭代并进行五次算法运行得到的运算结果可视化图表如下，其中第n个颜色的加入依据已有瓷砖颜色与已加入的$(n-1)$种颜色进行运算估计。

颜色加入序号	结果 1	结果 2	结果 3	结果 4	结果 5
1	98, 199, 164	162, 223, 110	103, 222, 158	129, 33, 32	174, 36, 223
2	93, 200, 164	150, 64, 223	197, 60, 191	95, 222, 159	192, 222, 106
3	220, 53, 222	225, 64, 223	113, 104, 218	217, 78, 162	221, 52, 223
4	195, 212, 112	119, 144, 144	33, 32, 80	192, 223, 92	43, 32, 223
5	101, 144, 223	128, 84, 223	32, 34, 87	213, 199, 116	209, 47, 61
6	209, 47, 61	193, 212, 95	96, 157, 207	205, 31, 54	223, 64, 65
7	118, 144, 146	214, 50, 60	164, 47, 107	212, 47, 59	223, 84, 143
8	52, 30, 57	223, 96, 146	108, 139, 163	105, 167, 225	94, 144, 153
9	175, 63, 139	178, 75, 155	32, 177, 135	196, 76, 148	52, 30, 57
10	113, 142, 141	129, 154, 140	46, 49, 43	146, 117, 144	47, 56, 33

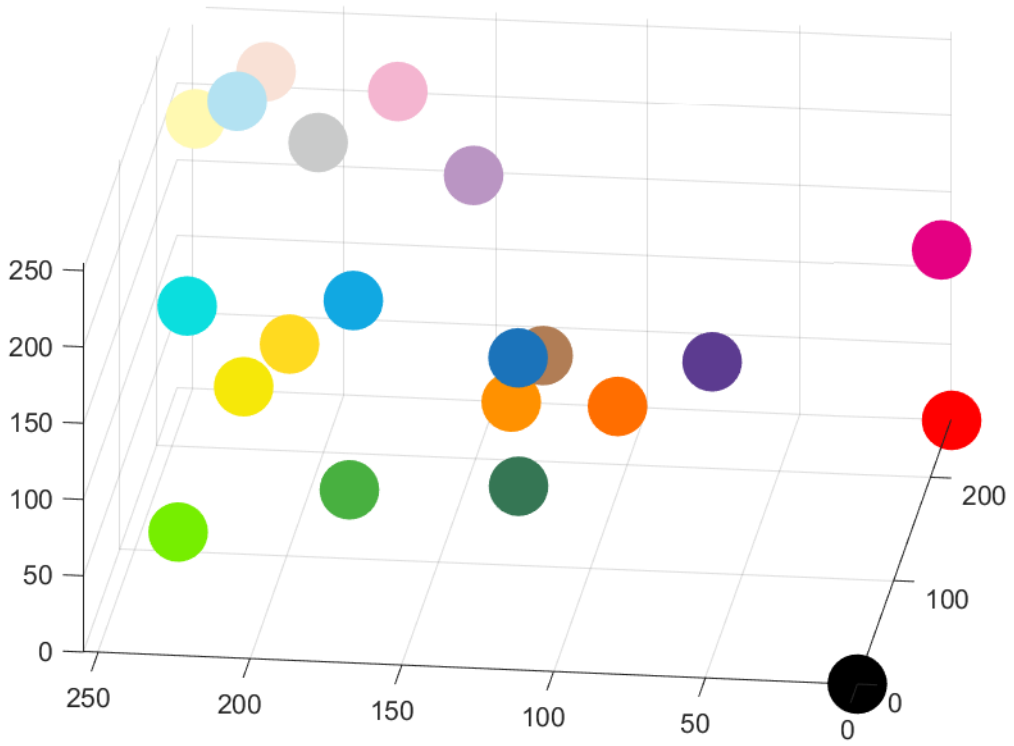
最终得到为提高图像表现力需要增加的十种颜色与RGB值的图表如下：

R	G	B	
129	33	32	Color23
95	222	159	Color24
221	52	223	Color25
43	32	223	Color26
128	84	223	Color27
193	212	95	Color28

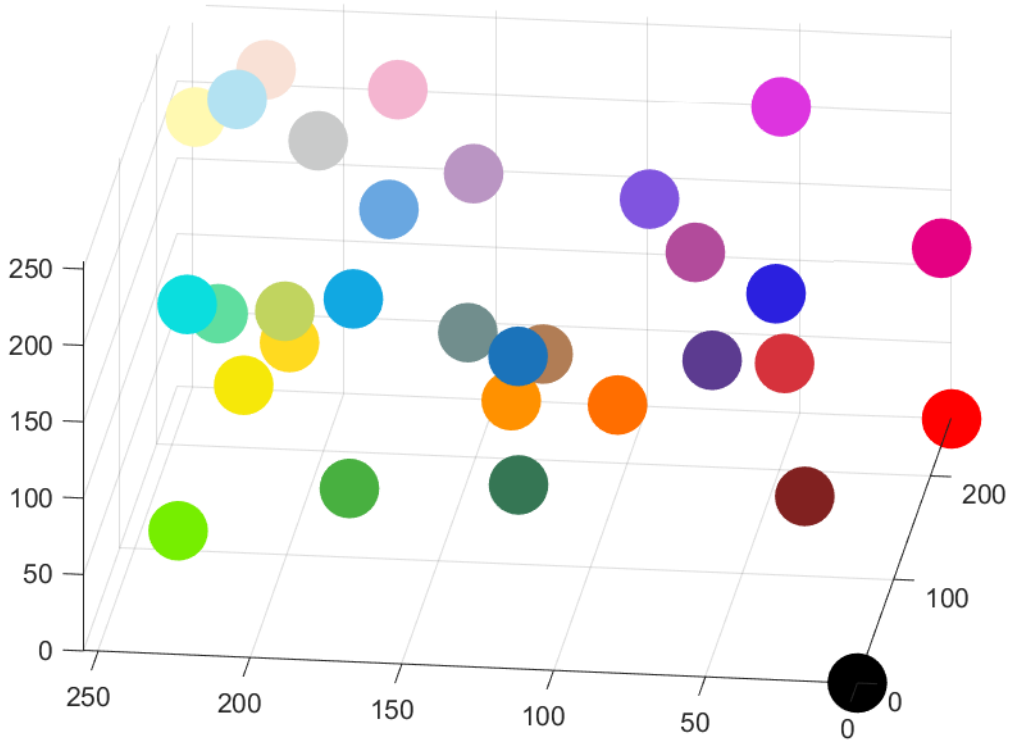
214	50	60	Color29
105	167	225	Color30
178	75	155	Color31
113	142	141	Color32

同时对于之前现有的 22 种RGB颜色空间, 与已经添加了十种颜色的RGB颜色空间对比, 可视化结果如下图:

（原有瓷砖颜色在空间中的分布）



（原有瓷砖颜色加十种颜色在空间中的分布）



5. 2. 4 模型结果分析

在结果中适应度函数的最大值逐渐下降，但下降程度很少，说明了贪心算法的合理性，对于出现多种颜色均达到适应度函数的最大值，符合实际情况。我们选取五次结果中适应度函数值最大的颜色作为新添加的颜色，从可视化结果分析，新得到的颜色与已有的颜色在人工观察下相差较大，可知遗传算法模型得到的新添加的颜色是较为合理的 同时对于之前现有的 22 种RGB颜色空间，与已经添加了 3 种新的颜色的RGB颜色空间对比，可视化结果如下图：从颜色空间分布图的对比上来看，新添加的颜色使所有颜色在RGB颜色空间中的分布更加的均匀，符合模型求解的目标

5. 3 问题三模型的建立与求解

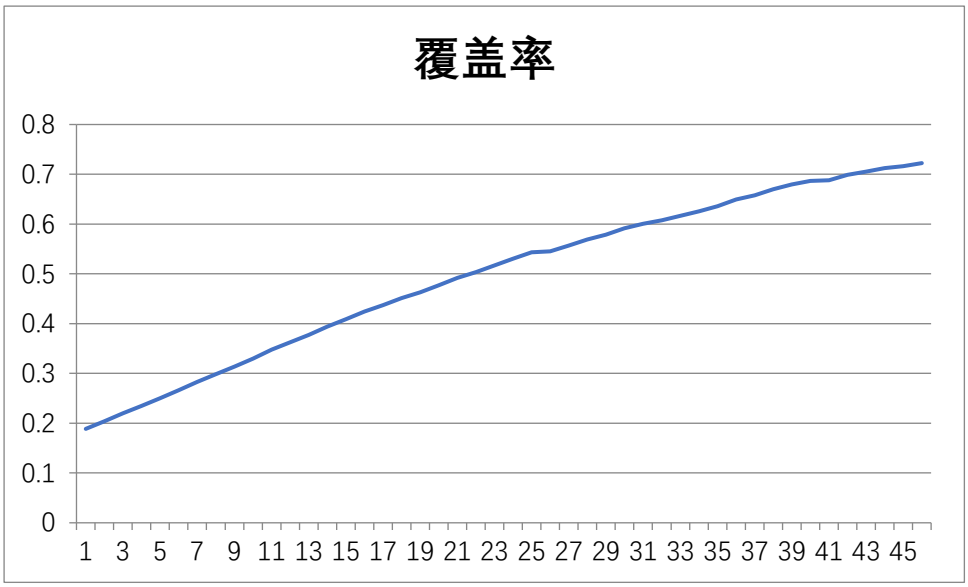
5. 3. 1 模型建立

在同时考虑成本和表现效果的情况下，我们借助优劣解距离法标准化两个评价量，其中w_1表示研发成本在评价过程中占有的比重，w_2表示图像展示效果在评价过程中所占有的比重，$w_1 \in [0,1]$，$w_2 = 1 - w_1$;

在检验颜色表现效果过程中，我们采用已有瓷砖颜色和新增瓷砖颜色对于整个颜色空间的覆盖率作为评价指标，并采用蒙特卡罗法对覆盖率进行标定，即：首先依据问题构造概率统计模型，再给出模型中各种不同分布随机变量的抽样方法，最后统计处理模拟结果，给出问题解的估计值。在评价覆盖率的过程中，我们需要通过蒙特卡罗法获取随机数检验瓷砖颜色的覆盖率。

对于每种既定颜色RGB值在三维坐标下的覆盖体积我们按照40的适用半径，即$r = 40$;同时结合色彩学和实际情况，我们将 80%的覆盖率视作覆盖上界，令当前覆盖率为q，则表现效果为$\frac{q}{80\%}$。根据常识我们可推得选取的颜色越多，瓷砖颜色的覆盖率越好，但受到瓷砖

颜色研发成本的限制，我们对增加颜色的数量设定上限。在我们之后对颜色覆盖率的量化分析后发现，在增加颜色数量小于 45 时，覆盖率与增加数量大致呈正比趋势，但在增加数量超 45 之后，覆盖率增加的趋势并不明显，故我们限定增加颜色的上限为 45 个。

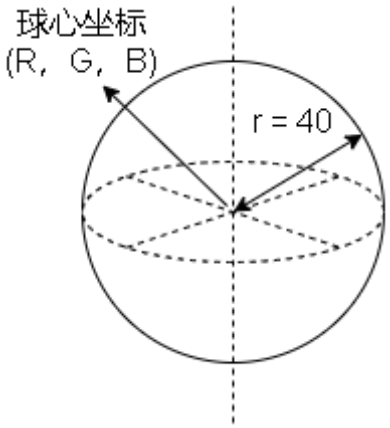


我们的求解过程如下：

$Step1$:结合第二问通过遗传算法选取的前n种颜色（$1 \le n \le 45$）；

$Step2$:原有瓷砖的 22 种颜色与选取的颜色组成集合；

$Step3$:集合中颜色的 RGB 值对应三维空间的坐标，由于规定颜色的适用半径$r = 40$，故每个颜色的覆盖空间为以集合 RGB 值为圆心、以适用半径为半径的球体（暂时不考虑球体重合情况）；



$Step4$:利用蒙特卡罗法对集合颜色覆盖空间的覆盖率进行模拟处理，最后经计算得到表现效果值；

$Step5$:对于成本量的评价，由于每一个新颜色的瓷砖的成本是相同的，所以我们选择颜色瓷砖的个数来衡量成本量，基础的 22 个颜色瓷砖为成本量最小值，以覆盖率达到一定需求的大小，而且不再明显线性增长时新添加的颜色个数加 22 个基础颜色个数为成本量的最大值；

$Step6$:借助 TOPSIS 优劣解距离法的思想来对成本和表现效果进行综合评价，先对成本做正向化处理，成本量为转化为极大值指标，公式如下：

$$m_{(极大化指标)} = max - m_{(极小化指标)}$$

再对成本量与表现效果这两个指标做标准化处理，由这两个评价对象和两个评价指标构成的正向化矩阵如下：：

$$M=\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$$

其标准化的矩阵T的每一个元素$t_{ij}=\frac{m_{ij}}{\sqrt{\sum_{i=1}^{2}{x_{ij}}^2}}$，即：

$$T=\begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix}$$

标准化两个指标之后，再通过计算评分的公式对于两个指标进行评分，评分过程如下：

定义最大值 $T^+=(T_1^+,T_2^+)=(max\{t_{11},t_{21}\},max\{t_{12},t_{22}\})$;

定义最小值 $T^-=(T_1^-,T_2^-)=(min\{t_{11},t_{21}\},min\{t_{12},t_{22}\})$;

定义第$i(i=1,2)$个评价对象与最大值的距离$D_i^+=\sqrt{\sum_{j=1}^{m}w_j(T_j^+-t_{ij})^2}$;

定义第$i(i=1,2)$个评价对象与最小值的距离$D_i^-=\sqrt{\sum_{j=1}^{m}w_j(T_j^--t_{ij})^2}$;

我们可以计算得到第$i(i=1,2)$个评价对象未归一化的得分：$S_i=\frac{D_i^-}{D_i^++D_i^-}$;

其中$0\le S_i\le 1$，且S_i越大D_i^+越小，即越接近最大值;

我们可以将得分归一化：$\widetilde{S_i}=\frac{S_i}{\sum_{i=1}^{2}S_i}$，其中$\sum_{i=1}^{2}\widetilde{S_i}$=1

本文首先赋予了成本量与表现效果的权重$w_1:w_2=1:1$，选出综合评价最高的方案。

$Step7$:对于权重系数我们进行灵敏度分析，来判断侧重减少成本，与侧重增大色彩表现力时应该选择哪种方案，即增加什么数量和什么颜色。

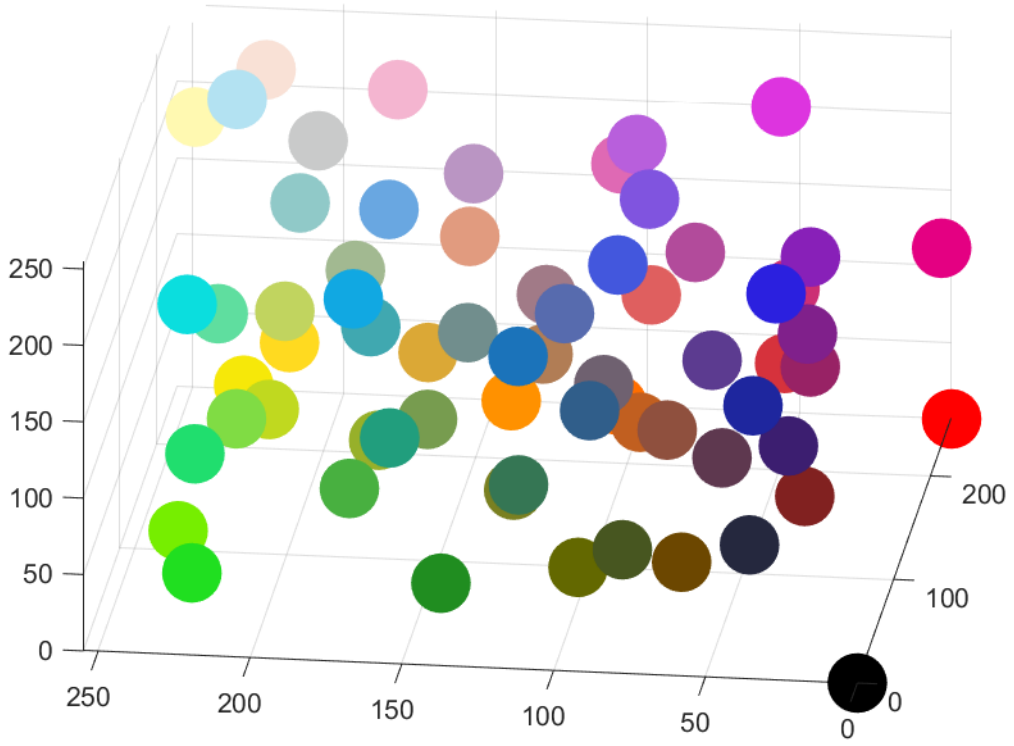
5. 3. 2 模型求解

依据第二问的遗传算法，我们求得为提高图像表现力需要增加的四十五种颜色与 RGB 值的图表如下：

R	G	B	
129	33	32	Color23
95	222	159	Color24
221	52	223	Color25
43	32	223	Color26
128	84	223	Color27
193	212	95	Color28
214	50	60	Color29
105	167	225	Color30
178	75	155	Color31
113	142	141	Color32
37	40	62	Color33
122	128	33	Color34
32	222	110	Color35
152	34	101	Color36

128	220	68	Color37
219	168	54	Color38
30	38	158	Color39
64	168	176	Color40
32	141	32	Color41
225	155	127	Color42
67	87	221	Color43
162	185	145	Color44
136	32	184	Color45
71	86	32	Color46
108	71	78	Color47
94	56	79	Color48
33	158	125	Color49
48	94	138	Color50
223	105	180	Color51
144	201	200	Color52
204	47	116	Color53
184	95	220	Color54
143	80	62	Color55
223	95	95	Color56
32	223	32	Color57
60	30	112	Color58
192	217	31	Color59
151	176	42	Color60
161	122	135	Color61
99	104	179	Color62
193	95	32	Color63
87	107	174	Color64
119	156	79	Color65
128	32	139	Color66
111	97	112	Color67

原有二十二种瓷砖颜色与新增四十五种颜色在空间中的分布如图:

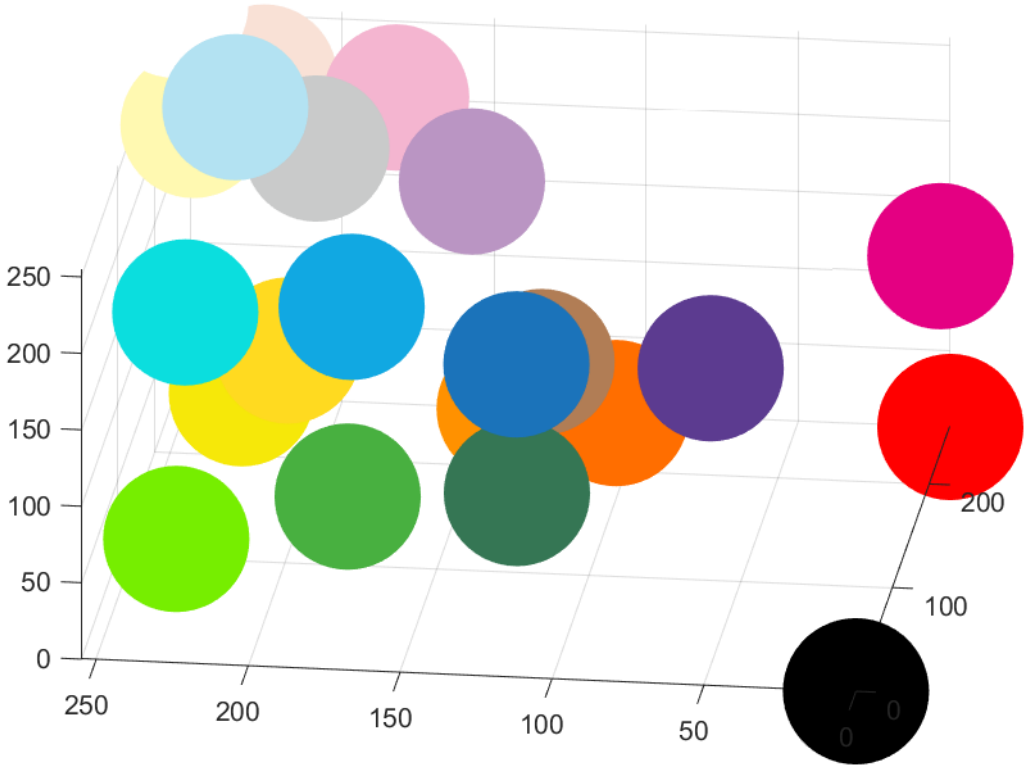


通过蒙特卡罗法对原有瓷砖颜色和新增颜色的覆盖率模拟估计得到图表结果如下:

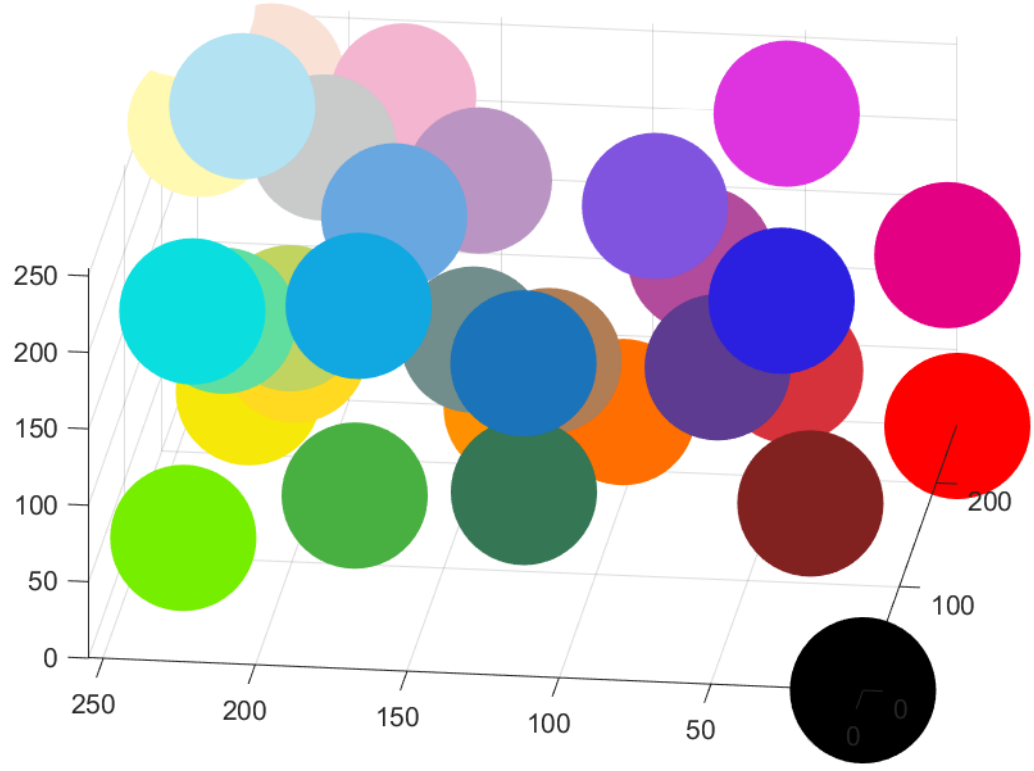
新增色数	覆盖率	新增色数	覆盖率
0	0.188641	23	0.530564
1	0.204009	24	0.543656
2	0.219814	25	0.545214
3	0.234524	26	0.556761
4	0.250091	27	0.569191
5	0.266087	28	0.578631
6	0.283006	29	0.591513
7	0.298542	30	0.600428
8	0.313902	31	0.607816
9	0.329987	32	0.616865
10	0.347955	33	0.625936
11	0.362405	34	0.635826
12	0.377489	35	0.649651
13	0.393876	36	0.657971
14	0.409164	37	0.669705
15	0.42432	38	0.679443
16	0.437195	39	0.686953
17	0.451411	40	0.687751
18	0.463103	41	0.699011
19	0.477358	42	0.705523
20	0.49195	43	0.712695
21	0.503658	44	0.716609
22	0.517062	45	0.72247

我们令颜色的适用半径$r = 40$, 将瓷砖颜色与新加颜色在空间中的覆盖率以散点图的形式可视化表示出来，图像如下：

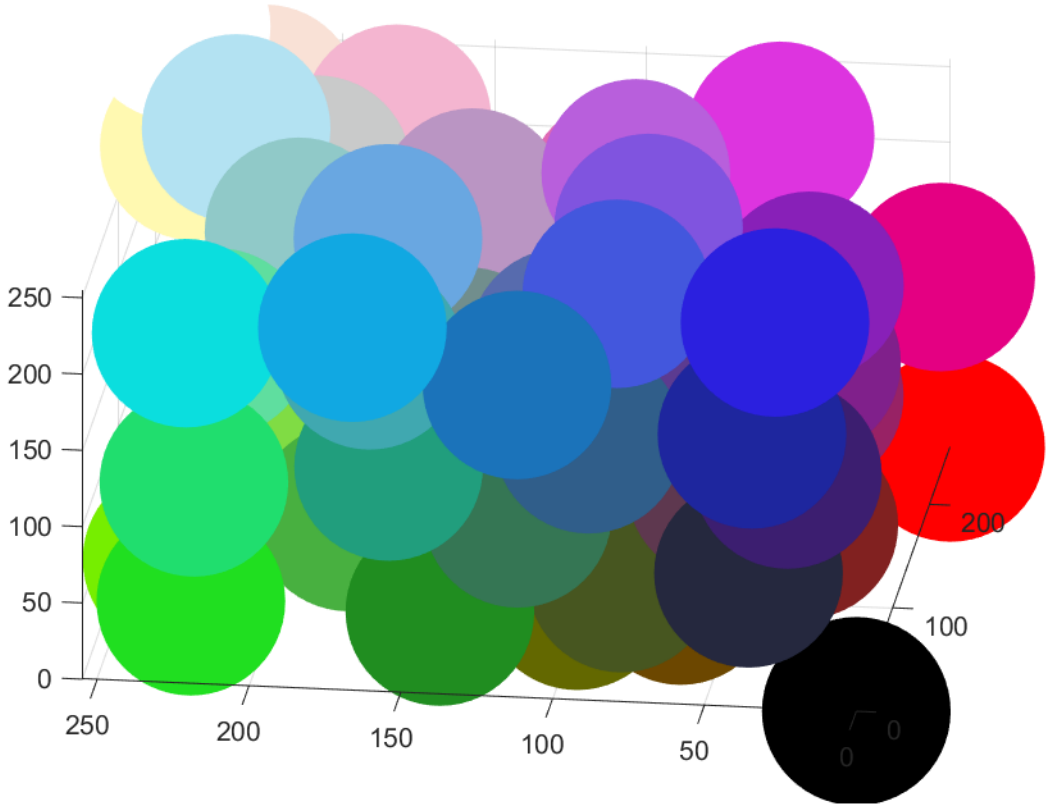
（原有瓷砖颜色在空间中的覆盖率）



（原有瓷砖颜色加新增十种颜色在空间中的覆盖率）



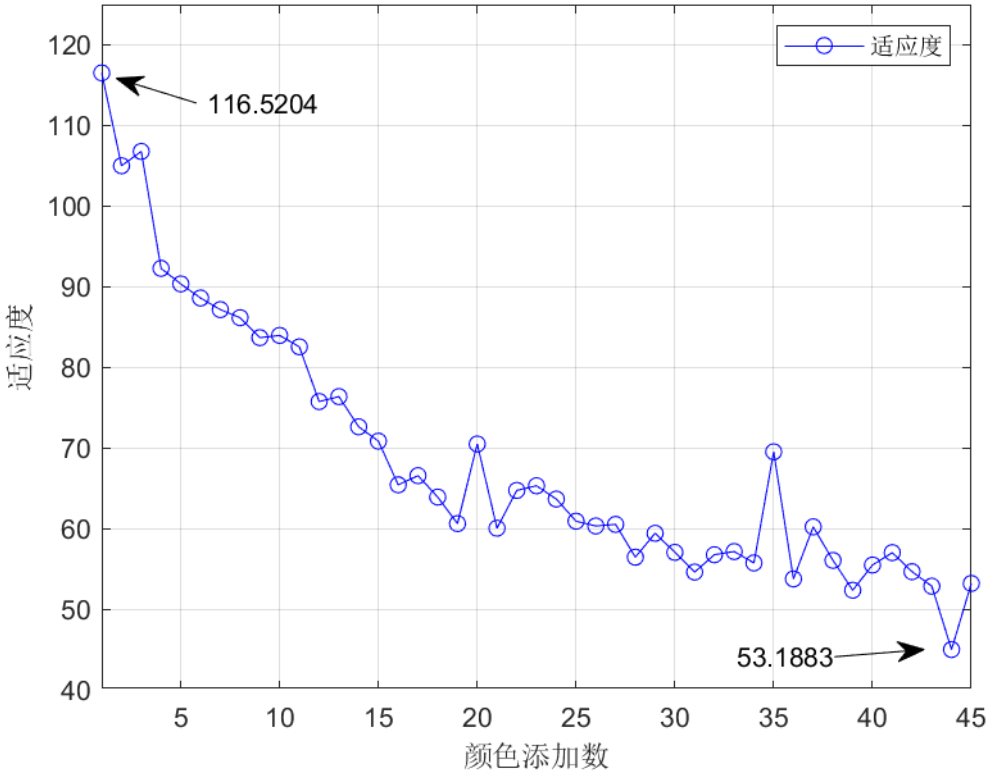
（原有瓷砖颜色加新增四十五种颜色在空间中的覆盖率）



对添加颜色的适应度进行分析，结果如下：

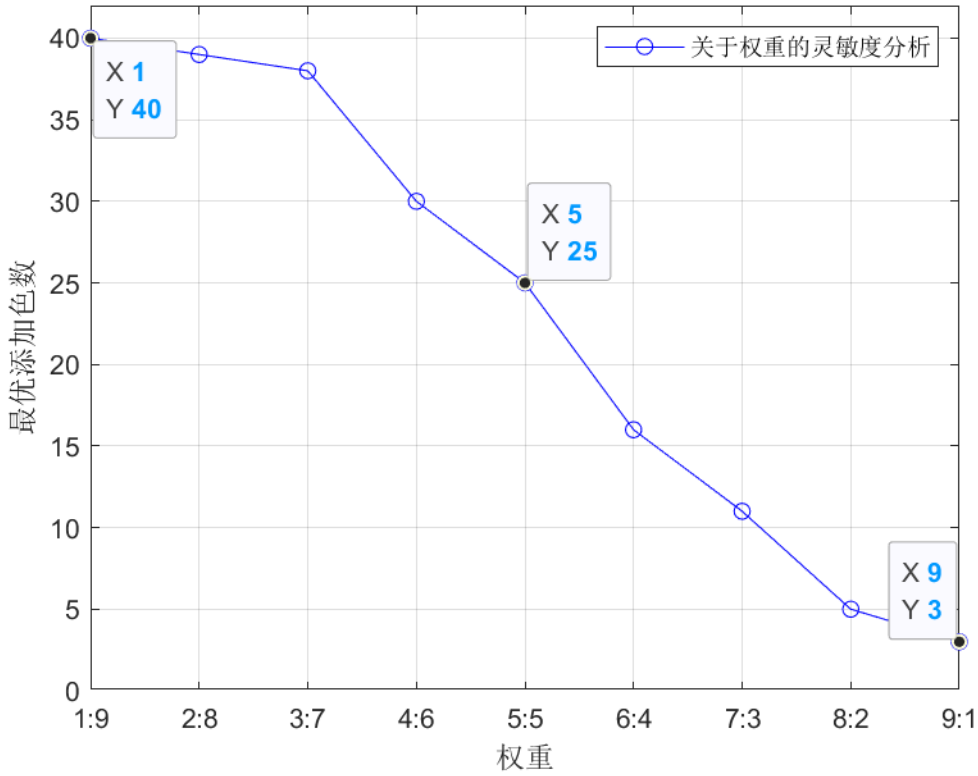
添色数	适应度	添色数	适应度
1	116.520384	24	63.65532185
2	105.000000	25	60.90976933
3	106.780148	26	60.3158354
4	92.271339	27	60.51446108
5	90.343788	28	56.44466317
6	88.594582	29	59.41380311
7	87.155034	30	57.04384279
8	86.151030	31	54.59853478
9	83.677954	32	56.75385449
10	83.952367	33	57.14892825
11	82.540899	34	55.73149917
12	75.749587	35	69.51978136
13	76.367532	36	53.7494186
14	72.615425	37	60.20797289
15	70.837843	38	56.04462508
16	65.421709	39	52.34500931
17	66.558245	40	55.47071299
18	63.898357	41	57.00877125
19	60.613530	42	54.64430437

20	70.484041	43	52.84884105
21	60.041652	44	44.98888752
22	64.730209	45	53.18834459
23	65.306967		



首先我们赋予了成本量与表现效果的权重为$w_1:w_2=1:1$，即权重为 5:5，我们对权重进行不断调整，获得在不同权重的条件下，最应该增加的颜色数量，结果如图：

权重	最优添色数
1 : 9	40
2 : 8	39
3 : 7	38
4 : 6	30
5 : 5	25
6 : 4	16
7 : 3	11
8 : 2	5
9 : 1	3



综合上述结果，在一般情况下同时考虑成本与表现效果，其权重相同，本文建议选择新添加前 25 种颜色，所选颜色如下：

R	G	B	
129	33	32	Color23
95	222	159	Color24
221	52	223	Color25
43	32	223	Color26
128	84	223	Color27
193	212	95	Color28
214	50	60	Color29
105	167	225	Color30
178	75	155	Color31
113	142	141	Color32
37	40	62	Color33
122	128	33	Color34
32	222	110	Color35
152	34	101	Color36
128	220	68	Color37
219	168	54	Color38
30	38	158	Color39
64	168	176	Color40
32	141	32	Color41
225	155	127	Color42

67	87	221	Color43
162	185	145	Color44
136	32	184	Color45
71	86	32	Color46
108	71	78	Color47

5. 3. 3 模型结果分析

本模型通过 TOPSIS 优劣解距离法的思想对成本与表现效果进行综合评价，通过检验结果可知选取新添加 25 种颜色是较为合理的，这时成本增加了研发 25 种颜色的成本，而其色域的覆盖已经可以到达了 0.545 是较为理想的情况。

而同时我们通过灵敏度分析可知当需求侧侧重于增大表现效果时候，可以将表现效果的权重则增大，可以选择更多的颜色。如当需求的表现效果与成本的权重为 7：3 时可以选取新添加 38 种颜色，因为权重更大时增加的颜色数变少，表现效果的增幅变得很小，不需要再进行选取新颜色，这时其覆盖率达到了 0.679，效果明显。

六、 模型的评价、改进与推广

6. 1 模型的优点

（1）本文第一问的近似性度量函数，经过检验具有较强的适用性，对于提供的马赛克瓷砖可以选择相似的颜色瓷砖来进行代替，忽略了标准差的影响也使近似性判断更加的简单直接，减少权重未知带来的误差与影响。

（2）本文第二问建立遗传算法模型，不仅符合了RGB空间的三维坐标数量级，而且符合RGB坐标的整数特点，可以快速地找到最优解。遗传算法中的适应度函数选取合理，可以找到使其RGB颜色空间分布更加均匀的新颜色。

（3）本文第三问的贪心算法，从结果层面分析，每一步得到的最优适应度函数值相差不大，说明贪心算法的选择合理，即一次一次地新添加颜色与同时添加多种新颜色结果相差很小，模型设计合理。

（4）本文第四问的蒙特卡罗模拟，可以很好的计算已选颜色的覆盖率，来对于表现效果进行评价，成本与表现效果之间的权重我们通过灵敏度分析来考虑，为厂商提过更好地选择。

6. 2 模型的缺点

本文假定RGB颜色空间分布均匀，但在实际应用中，受人眼的识别程度和视感的限制，简单量化并不能准确反映颜色差异。通过对RGB颜色空间及色差度量公式的研究，人眼对RGB三种变量的敏感程度不同；两种颜色的色差受两者空间距离与矢量角度的影响；RGB的三个分量在不同情况下重要程度不同。仅考虑空间距离以及仅计算两颜色在RGB空间中的欧氏距离是不够的。

6. 3 模型的改进

（1）本文的模型在第一问中，省略标准差的影响，模型改进时可以考虑通过综合考虑

欧氏距离与标准差，通过相关权重来构造相似性度量的函数。或者引用最新的 RGB 色差公式[4]，由于三原色的分量三者并不独立，故需要对红黄蓝三原色按照人眼对其的敏感程度进行加权，以降低色差提高颜色准确度。

（2）本文采用了RGB颜色空间的度量，模型改进时是可以考虑转化为*HSV*颜色空间进行相似性的度量，相较于RGB空间，*HSV*空间能够非常直观的表达色彩的明暗，色调，以及鲜艳程度，使用*HSV*方法能更方便进行颜色之间的对比与RGB颜色空间进行对比来改进马赛克瓷砖的选取。

6.4 模型的推广

我们对马赛克瓷砖颜色的算法模型对数字油画、钉子画、钻石画等新型 DIY 创意产品提供了借鉴意义。这些 DIY 创意产品受顾客专业水平的限制，并不能为其提供所有颜色区域的分布情况，而我们的算法为厂商解决如何以最高效在最大程度上满足顾客的需求，这在一定程度上推动制造业和服务业的发展。

七、 参考文献

[1]陈抒瑢. 基于特征提取和色彩分割的偏色检测[D].南京大学,2014.

[2]王文洪,刘洪江.一种小色差瓷砖颜色自动分类器设计[J].海峡科技与产业,2017(08):151-153.

[3]张诗煜,嵇启春,孟月波.基于黄金分割双种群遗传算法的区域交通信号控制[J/OL].计算机测量与控制:1-8[2021-06-04].

[4]杨振亚,王勇,杨振东,王成道.RGB 颜色空间的矢量-角度距离色差公式[J].计算机工程与应用,2010,46(06):154-156.

附录

附录 1

第一问代码

```
clear;
load('base.mat')
load('graph1.mat')
load('graph2.mat')
len1 = length(graph1);
len2 = length(graph2);
opt_Eu1 = zeros(len1, 1);
opt_Eu2 = zeros(len2, 1);

optRGB_Eu1 = zeros(len1, 3);
optRGB_Eu2 = zeros(len2, 3);

%% 欧式距离
for i = 1 : len1
    mindist = 1000000000;
```

```
    for j = 1 : 22
        dist = (graph1(i, 1) - base(j, 1)) ^ 2 + (graph1(i, 2) -
base(j, 2)) ^ 2 + (graph1(i, 3) - base(j, 3)) ^ 2;
        if dist < mindist
            mindist = dist;
            opt_Eu1(i) = j;
            optRGB_Eu1(i, :) = base(j, :);
        end
    end
end

for i = 1 : len2
    mindist = 1000000000;
    for j = 1 : 22
        dist = (graph2(i, 1) - base(j, 1)) ^ 2 + (graph2(i, 2) -
base(j, 2)) ^ 2 + (graph2(i, 3) - base(j, 3)) ^ 2;
        if dist < mindist
            mindist = dist;
            opt_Eu2(i) = j;
        end
    end
end
```

附录 2

适应度函数代码

```
function fitness=fitnessFunc(chromosome)
%% 选择该色到 set 色集的所有点距离的最小值作为个体的适应度
global set
%% 计算染色体表现型 24 个 2 进制数 转 3 个十进制数 xyz 坐标
len=length(chromosome);
numList=2.^(len / 3 -1 : -1 : 0);   %二进制权重表
x = sum(chromosome(1 : 8) .* numList);
y = sum(chromosome(9 : 16) .* numList);
z = sum(chromosome(17 : 24) .* numList);
r = 30;
% 若点落在在边界, 不纳入考虑
if abs(x - 0) < r || abs(x - 255) < r || abs(y - 0) < r || abs(y -
255) < r || abs(z - 0) < r || abs(z - 255) < r
    fitness = -1;
else
    % 计算表现型对应的适应度
    lenBase = length(set);
    dist = zeros(lenBase, 1);
```

```
    % 选择该色到 set 色集的所有点距离的最小值作为个体的适应度
    for i = 1 : lenBase
        dist(i, 1) = sqrt((x - set(i, 1))^2 + (y - set(i, 2))^2 +
(z - set(i, 3))^2);
    end
    fitness = min(dist);
end

end
```

附录 3

遗传算法代码

```
function
[bestChromosome,fitnessBest]=GA(numOfChromosome,numOfGene,iteration
Num)
%% 随机生成初始种群，种群大小为 numOfChromosome，染色体中基因数为
numOfGene
% 进度条
bar = waitbar(0,'进度条')
lastPopulation = randi([0, 1], numOfChromosome, numOfGene);
newPopulation = zeros(numOfChromosome, numOfGene);

%% 最大迭代次数 iterationNum
for iteration = 1 : iterationNum
    waitbar(iteration / iterationNum,bar)
    %% 计算所有 numOfChromosome 个个体（染色体）的适应度值
    fitnessAll = zeros(1,numOfChromosome);
    for i = 1 : numOfChromosome
        individual = lastPopulation(i,:);
        fitnessAll(i) = fitnessFunc(individual);
    end

    %% 如果达到最大迭代次数，跳出变异遗传
    if iteration == iterationNum
        break;
    end

    %% 使用轮盘赌法选择 numOfChromosome 条染色体，种群中个体总数不变
    fitnessSum = sum(fitnessAll);
    fitnessProportion = fitnessAll / fitnessSum;
    % 使用随机数进行 numOfChromosome 次选择，保持种群中个体数量不变
    for i=1 : numOfChromosome
        probability = rand(1);
```

```
        proportionSum = 0;
        chromosomeIndex = 1;
        for j = 1 : numOfChromosome
            proportionSum = proportionSum + fitnessProportion(j);
            if proportionSum >= probability
                chromosomeIndex = j;
                break;
            end
        end
        newPopulation(i,:) = lastPopulation(chromosomeIndex, : );
    end

    %% 将染色体进行配对，执行单点交叉
    lastPopulation=newPopulation;
    % 生成从 1 到 numOfChromosome 的无序排列，每两个相邻数字进行配对
    coupleAllIndex = randperm(numOfChromosome);
    for i=1:floor(numOfChromosome/2)
        coupleOneIndex=coupleAllIndex(2*i-1:2*i);
        % 两条染色体交叉的概率
        probability=0.5;
        if rand(i)<probability
            crossPosition=randi([1,numOfGene],1);


newPopulation(coupleOneIndex(1),crossPosition:end)=lastPopulation(c
oupleOneIndex(2),crossPosition:end);

newPopulation(coupleOneIndex(2),crossPosition:end)=lastPopulation(c
oupleOneIndex(1),crossPosition:end);
        end
    end

    %% 对每条染色体执行基本位变异操作
    lastPopulation = newPopulation;
    for i = 1:numOfChromosome
        % 染色体变异的概率
        probability = 0.2;
        if rand(i) < probability
            mutatePosition = randi([1, numOfGene], 1);

            % 将对应基因位置的二进制数反转
            if(lastPopulation(i,mutatePosition) == 0)
                newPopulation(i,mutatePosition) = 1;
            else
```

```
                newPopulation(i,mutatePosition) = 0;
            end
        end
    end

    %% 迭代完毕，更新种群
    lastPopulation = newPopulation;
end

%% 遗传迭代结束后，获得最优适应度值和对应的基因
fitnessBest = max(fitnessAll);
bestChromosome = newPopulation(find(fitnessAll == fitnessBest,
1), : );
end
```

附录 4

第二问代码

```
clear;
global set
load("base.mat", "base");
load("raw.mat", "raw");
load("addition.mat", "addition");
%%参数设置区     set 为计算对象  addition:+45 色  base+10 色  raw+0
色
set = addition;
numOfChromosome = 100;
numOfGene = 24;
iterationNum = 15000;
[bestChromosome, fitnessBest] =
GA(numOfChromosome,numOfGene,iterationNum);

numList=2.^(7 : -1 : 0);    %二进制权重表
x = sum(bestChromosome(1 : 8) .* numList);
y = sum(bestChromosome(9 : 16) .* numList);
z = sum(bestChromosome(17 : 24) .* numList);
```

附录 5

蒙特卡罗法代码

```
function[vm] = mont_Coverage(times, r, set, rbound)
 %% 计算给定参数条件下的蒙特卡洛对于 set 色集在 r 下命中率
hit = 0;
for i = 1 : times
```

```
    x = randi([1, 255], 1);
    y = randi([1, 255], 1);
    z = randi([1, 255], 1);

%     lenBase = length(set);
    lenBase = rbound;
    flag = 0;
    for j = 1 : lenBase
        dist = sqrt((x - set(j, 1))^2 + (y - set(j, 2))^2 + (z -
set(j, 3))^2);
        if dist <= r
            flag = 1;
            break;
        end
    end
    if flag == 1
        hit = hit + 1;
    end
end
vm = hit / times;
 end


clc, clear;
load("base.mat", "base");
load("raw.mat", "raw");
load("addition.mat", "addition");
load("coverage.mat", "coverage");
res = []

%%参数设置区      set 为计算对象  addition:+45 色  base+10 色  raw+0
色
times = 1000000;
r = 40;
set = addition;

for rbound = 22 : 67
    t = mont_Coverage(times, r, set, rbound);
    res = [res; t]
end

% plot(res, 'bo-');
```

附录 6

第三问代码

```
clear;clc
load X.mat
Sensitivity_analysis = []

%%  初始化参数
[n,m] = size(X);

% disp([ num2str(n) '个评价项目, ' num2str(m) '个评价指标'])

%% 评判矩阵已正向化,直接进行标准化
Z = X ./ repmat(sum(X.*X) .^ 0.5, n, 1);

%% 枚举权重
for i = 1 : 9
% 设置权重
    weight = [i, 10 - i] / 10;

% 计算与最大值的距离和最小值的距离，并算出得分
    D_P = sum([(Z - repmat(max(Z),n,1)) .^ 2 ] .*
repmat(weight,n,1) ,2) .^ 0.5;   % D+ 与最大值的距离向量
    D_N = sum([(Z - repmat(min(Z),n,1)) .^ 2 ] .*
repmat(weight,n,1) ,2) .^ 0.5;   % D- 与最小值的距离向量
    S = D_N ./ (D_P+D_N);     % 未归一化的得分
    stand_S = S / sum(S)
    [sorted_S,index] = sort(stand_S ,'descend')
    Sensitivity_analysis = [Sensitivity_analysis; [i, index(1,
1)]];
end
% plot(Sensitivity_analysis( : , 2), 'bo-')
```