

基于 HSV 色彩空间的瓷砖选色搜索算法

摘 要

在瓷砖选色问题中，如何判断两种颜色的接近程度和评价选色效果是问题的关键所在。本文就瓷砖选色问题，建立了一套基于 HSV 色彩空间的目标搜索和评价方法。

针对问题一，定义了颜色接近程度的判断方法和瓷砖选色搜索方法。为了符合人眼视觉的感知特性，我们将研究空间从不适合表现两种颜色的视觉差异的 RGB 色彩空间转移到 HSV 色彩空间：首先将图像 1、2 与瓷砖颜色 RGB 值通过色彩学定义转换为 HSV 值，再在 HSV 色彩空间中建立 XYZ 三维坐标系，将 HSV 值转换为对应空间中的 XYZ 三维坐标，至此，完成了研究空间的搭建。基于 HSV 色彩与人感受紧密相连的特性，我们认为欧氏距离可以代表此情况下瓷砖选色时的颜色接近程度，类比索引图像概念，我们将瓷砖颜色列表作为调色盘，基于欧氏距离构建了针对图像 1、2 的循环搜索算法。最终，我们通过 python 工具完成了基于 HSV 色彩空间的最小欧氏距离循环搜索算法，得出了最接近图像颜色的瓷砖选色结果，保存在附件 result1.txt 和 result2.txt 中。

针对问题二，定义了对新增颜色的评价打分公式，并建立了非线性规划模型选择最优新增瓷砖颜色。首先，由于该问题要求在提升拼接表现力的条件下，进行最优新增颜色选择，因此我们不能简单的采用欧氏距离评价颜色接近程度的方法，基于此种情况，我们定义了一种对新增颜色打分的公式——HSV 欧氏距离，该公式基于权重法的思想，突出了色彩的 H、S 色调分量，降低了 V 亮度分量的影响，以此选择更多彩的新增颜色。同时，我们依旧在 HSV 色彩空间中寻找最优新增颜色，借助 HSV 分布于圆柱体的特性，可以将该寻优问题变为非线性规划模型解决。由于非线性规划模型存在算法容易陷入“局部最优解”的问题，因此在该问题中本文选择了多种算法进行对比分析，找出最佳算法，尝试逃出局部最优，寻找“全局最优解”。本文选择了蒙特卡罗模拟(MCS)、基于蒙特卡罗的内点法搜索(MCSIPM)、遗传算法搜索(GA)，最终发现蒙特卡罗在本问题中能找到更优的解。最终，我们通过 Matlab 工具，在 HSV 空间中，建立了一套基于打分评价的最优新增颜色选择循环模拟算法，选出了同时新增 1-10 种瓷砖情况下的最优瓷砖颜色选择结果，并将结果转换回 RGB 编码值保存。

针对问题三，定义了一种对表现效果的度量公式，并构建了寻找最优成本效果点的最优化模型。首先，本文针对瓷砖点的表现效果理解为与空间分布的均匀程度有关，同时结合问题二的思考，定义了一种新度量——色彩平面立体综合距离，基于这种新度量，定义了 F 函数，并通过蒙特卡罗概率模型求得 F 函数值。借助 F 函数值，由增长率定义了评估函数 1，由成本定义了评估函数 2。至此，将题目的“综合考虑成本和表现效果”转换为“综合评估函数 1 和评估函数 2 结果”，采用权重法，得出综合评估指标 $E=5.1645$ ，根据该指标，我们确定应该新增 5 种瓷砖颜色，颜色选择与问题二的新增 5 种颜色相同。

本文的亮点在于：首先，将研究空间定义在符合人眼视觉感知特性的 HSV 空间；其次，在非线性规划模型中采用了多种算法对比，并选择最优算法；另外，本文由问题一基于欧氏距离评价颜色接近程度，改进出问题二的 HSV 欧氏距离和问题三的色彩平面立体综合距离，并有很多新定义进行评价，如 F 函数，评估函数等。

关键词：HSV 色彩空间；欧氏距离；蒙特卡罗模拟；权重法；最优化模型

目录

基于 HSV 色彩空间的瓷砖选色搜索算法.....	1
一、 问题重述.....	3
1.1 问题背景.....	3
1.2 问题提出.....	3
二、 问题分析.....	3
2.1 问题一的分析.....	3
2.2 问题二的分析.....	3
2.3 问题三的分析.....	4
三、 模型假设.....	4
四、 符号说明.....	4
五、 模型的建立与求解.....	5
5.1 问题一的模型建立与求解.....	5
5.1.1 RGB 颜色值转化为 HSV 颜色值	5
5.1.2 HSV 颜色值转化为 XZY 坐标值	6
5.1.3 基于欧氏距离判断最接近瓷砖颜色.....	7
5.1.4 建立选色模型与求解.....	8
5.2 问题二的模型建立与求解.....	12
5.2.1 拼接图像表现力打分建模.....	12
5.2.2 非线性规划模型建立.....	12
5.2.3 最大距离求解.....	12
5.2.4 瓷砖颜色坐标转为 RGB 值	13
5.3 问题三的模型建立与求解.....	15
5.3.1 评价表现效果的新度量定义.....	15
5.3.2 基于蒙特卡罗概率模型的建立.....	16
5.3.3 评估函数 1——增长率定义.....	16
5.3.4 评估函数 2——成本定义.....	16
5.3.5 最佳成本效果点求解.....	17
5.3.6 综合指标评估.....	18
六、 模型的分析.....	19
6.1 问题一模型的分析.....	19
6.2 问题二模型的分析.....	19
6.3 问题三模型的分析.....	19
七、 模型的评价、改进与推广.....	20
7.1 问题一模型的评价、改进和推广.....	20
7.2 问题二模型的评价、改进和推广.....	20
7.3 问题三模型的评价、改进和推广.....	20
八、 参考文献.....	20
九、 附录.....	21

一、问题重述

1.1 问题背景

自上世纪八十年代以来,随着社会的发展,马赛克瓷砖的市场需求日益增长。马赛克瓷砖一般为边长不超过 5 厘米的正方形瓷砖,由于面积较小,所以可将其做成不同的样式,以不同的组合方式在平整的墙面或地板上绘制出文字图案。作为一种传统而有震撼力的装饰手段,马赛克成为了现代装饰中极具表现力的艺术形式。由于用于马赛克瓷砖的颜色无法覆盖所有的颜色种类,所以用户在采用马赛克瓷砖拼接图案时,需要首先挑选出与图案中的颜色相同或相近的马赛克瓷砖。

1.2 问题提出

某一家工厂受到设备及资金的限制,目前只能生产出 22 种纯色的马赛克瓷砖。由于彩色图像的颜色通常种类更多,因此就需要从 22 种颜色中,挑选出与彩色图像的颜色最接近的瓷砖。为了减少人工挑选瓷砖颜色的工作量,该厂决定开发一种自动根据用户提供图像信息,进行匹配相似颜色马赛克瓷砖的软件。其中图像均为 24bit 的 RGB 格式,现需根据提供的图像 RGB 颜色列表,通过建立数学模型,解决以下问题:

- (1) 确定根据原始图像搜索颜色接近程度最高的现有瓷砖颜色的算法,找出图像 1 和图像 2 每种颜色最接近的瓷砖颜色,并将选色的结果按格式输出保存。
- (2) 分析新增颜色对拼接图案表现力的影响,定义表现力的评价方法,选出应该优先研发的瓷砖颜色,输出同时增加 1、2、3...10 种颜色时对应的 10 种 RGB 颜色编号。
- (3) 各种颜色瓷砖的成本相同,通过定义表现效果,加上生产数量成本的惩罚限制,确定一种最佳的瓷砖颜色研发计划,确定研发瓷砖的数量,并给出相应颜色的 RGB 编号。

二、问题分析

2.1 问题一的分析

根据原图像的真彩色颜色列表找出相近的瓷砖颜色,可以把问题转化为在颜色空间中找到与图像颜色列表中的点距离最近的瓷砖颜色列表中的点,即把图像的颜色分量转化为三维坐标,通过循环计算点坐标之间的距离,找出最近的点。

由于图像包含的颜色种类多,计算量较大,考虑到速度优先的问题,我们采用欧氏距离计算点之间的距离。

但是需要注意的是, R/G/B 分量之间的相关性较强,导致在 RGB 颜色空间中的颜色连续的变化并不明显,所以在 RGB 颜色空间中用欧氏距离表示颜色在视觉感受上的最相似是不准确的。因此,我们需要一个基于更接近于人类感知经验建立的色彩空间(即 HSV 颜色模型)进行空间计算的三维坐标系。

综上,解决问题一的思路主要分为两步:(1) 将图像颜色的 RGB 分量转化为 HSV 分量,并在 HSV 色彩空间中建立 XYZ 三维坐标。(2) 对图像中每一种颜色循环每一块瓷砖,判断 HSV 空间中欧氏距离最短的点。针对问题一,结构化分块编写模型的代码并进行求解。

2.2 问题二的分析

24bit 的 RGB 图像所涵盖的颜色多达 2^{24} 种,如果想要用几种瓷砖颜色来代替 RGB 颜色的话,需要使这几种瓷砖颜色尽可能均匀的分布在 RGB 色彩中。那么考虑拼接图像的表现力开

发瓷砖，最重要的一点是要考虑到瓷砖的颜色尽可能地覆盖 RGB 图像含有的颜色，即考虑现有瓷砖颜色对 RGB 色彩的覆盖盲点。可基于欧氏距离改进一种对新增颜色打分的公式，利用蒙特卡罗方法，在 HSV 空间中随机寻找空间点，计算每个随机点与现有 22 个瓷砖颜色空间点的欧氏距离和，依此找到得分最大的点，找到的点即为最能提高已有颜色拼接表现力的点，再计算出该点颜色对应的 RGB 数值，便可以生产此种颜色的瓷砖。

从这个思路出发可以对不同颜色的表现效果进行打分，建立非线性规划模型，计算某一颜色到现有所有瓷砖颜色的改进欧氏距离和，距离和越大则表现效果越好，越应该增加该颜色。最后依次循环运行，并将前一次的结果作为后一次的新条件，得出同时增加 1 种、2 种、3 种.....颜色情况下改进欧氏距离和最大的 1 个、2 个、3 个.....10 个点，并计算出对应颜色的 RGB 值。

2.3 问题三的分析

问题三是在问题二得出结果的基础上，对同时增加 1 种、2 种、3 种.....颜色作成本和表现效果的综合分析，即考虑到成本就是增加的种类越少越好，考虑到表现成果就是增加的种类越多越好，针对这十种情况建立最优化目标规划模型，找出最优方案。

首先分析表现效果分量，定义一种突出 H 与 S 信息的综合距离，将问题二求解的结果和 22 个已经生产出的瓷砖颜色点的集合作为现有颜色点，放入 HSV 色彩空间中；接下来使用蒙特卡罗概率模型，在色彩空间中选取 10 000 000 个随机点，计算该点到现有颜色点的综合距离，若空间点到现有颜色点的综合距离小于 0.3，认为可以找到最接近颜色，符合条件。对该值定义一个函数，函数值越大，可以认为表现效果越好。接下来定义两个评估函数，评估函数 1 增长率，评估函数 2 成本，增长率为斜率相关函数，成本为惩罚分量，与增加瓷砖个数相关，每多生产一种颜色的瓷砖，就增加一次惩罚分量。将评估函数 1 和评估函数 2 进行权重分配，得到表现效果与增加的瓷砖个数的综合指标模型。从而求解出应该增加的瓷砖颜色个数。

三、模型假设

1. 在 HSV 色彩空间中，相邻的色彩就是颜色程度接近的色彩；
2. HSV 色彩空间为圆柱形，认为空间点具有唯一性；
3. 在选择瓷砖颜色的过程中，不考虑对特定图像的表现效果，只考虑通用表现效果；
4. 认为蒙特卡罗模拟得到的结果为最优结果；
5. 认为本文所定义的两改进欧氏距离和可以最佳地表现出拼接表现力和表现效果，认为其它新定义的函数表示正确。

四、符号说明

符号	说明	单位
h	色调	度
s	饱和度	/
v	明度	/
(x, y, z)	空间点的三维坐标	/
$d_m(x, y)$	m 维空间中两点之间的距离	/

续表

符号	说明	单位
d	三维空间中两点之间的距离	/
S_i	基于 HSV 色彩空间定义的 HSV 欧氏距离	/
S	HSV 欧氏距离和	/
D	度量色彩表现效果的定义, 即色彩 平面立体综合距离	/
F	增加瓷砖颜色个数的 符合条件率函数	/
$cost$	生产瓷砖颜色的成本惩罚分量	/
E	对增加瓷砖颜色不同数量的 综合考量指标	/
$evaluate$	关于增加不同数目颜色的评估函数	/

五、模型的建立与求解

5.1 问题一的模型建立与求解

5.1.1 RGB 颜色值转化为 HSV 颜色值

(1) 确定色彩空间从 RGB 转换为 HSV 的关系。设(r,g,b)为颜色的红、绿、蓝坐标, 取值范围为[0,1], (r,g,b)=(R,G,B)/255, max 为 r,g,b 中的最大值, min 为其中的最小值。HSV 色彩空间中有(h,s,v), 其中色调 $h \in [0, 360)$, 饱和度 $S \in [0, 1]$, 明度 $v \in [0, 1]$ 。公式如下:

$$h = \begin{cases} 0^\circ & \max = \min \\ 60^\circ \times \frac{g-b}{\max-\min} + 0^\circ & \max = r, g \geq b \\ 60^\circ \times \frac{g-b}{\max-\min} + 360^\circ & \max = r, g < b \\ 60^\circ \times \frac{g-b}{\max-\min} + 120^\circ & \max = g \\ 60^\circ \times \frac{g-b}{\max-\min} + 240^\circ & \max = b \end{cases}$$

$$s = \begin{cases} 0^\circ & \max = 0 \\ \frac{\max-\min}{\max} & \text{其他} \end{cases}$$

$$v = \max$$

(2) 将图像一颜色列表转换为 hsv。将 RGB 值按 216*3 的矩阵方式排列, 从 i=0 开始将每一行矩阵元素带入上述公式并循环上述操作, 至 i=216 退出循环, 返回值为一个 216*3 的矩阵 h11, 矩阵每一列元素依次为颜色的 h、s、v 的值。

(3) 将图像二颜色列表转换为 hsv。将 RGB 值按 216*3 的矩阵方式排列，从 i=0 开始将每一行矩阵元素带入上述公式并循环上述操作，至 i=200 退出循环，返回值为一个 200*3 的矩阵 hl2，矩阵每一列元素依次为颜色的 h、s、v 的值。

(4) 将现有瓷砖颜色列表转换为 hsv。将 RGB 值按 22*3 的矩阵方式排列，从 i=0 开始将每一行矩阵元素带入上述公式并循环上述操作，至 i=22 退出循环，返回值为一个 22*3 的矩阵 hc，矩阵每一列元素依次为颜色的 h、s、v 的值。

5.1.2 HSV 颜色值转化为 XYZ 坐标值

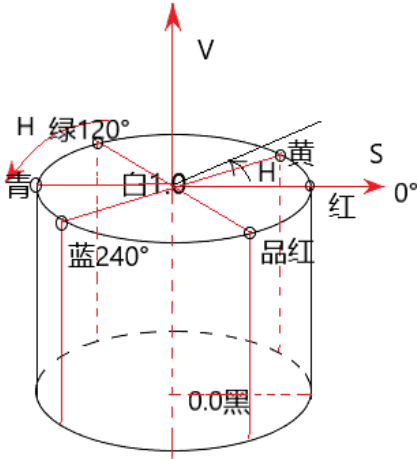


图 1 HSV 色彩空间模型

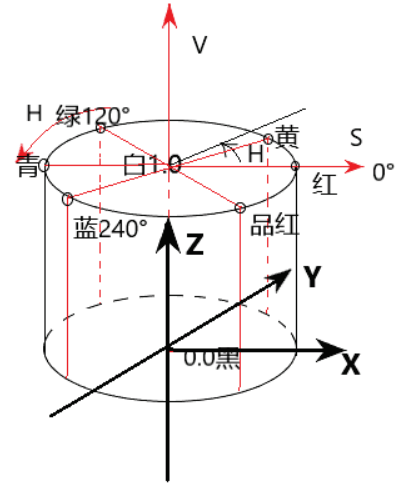


图 2 基于 HSV 的空间坐标系

HSV 色彩的空间模型为倒立的圆锥形，如图 1 所示，其中 H 为角度，S 为对应横切面圆的半径，V 为从底部顶点开始计算的高度，由这些数据不能直接计算欧氏距离，因此我们需要在 HSV 色彩空间中，建立三维坐标，求取对应的(X,Y,Z)坐标。

(1) 建立三维坐标系。以 HSV 圆锥体的底部顶点为原点，H=0 为 x 轴正方向，建立三维坐标系，如图 2 所示。

(2) 确定 hsv 和 xyz 的关系。s 和 v 取值范围都为[0,1]，色彩值(h,s,v)与三维坐标点(x,y,z)的对应关系简化如下：

$$\begin{cases} x = s \times \cos h \\ y = s \times \sin h \\ z = v \end{cases}$$

(3) 将图像一的 hsv 值转化为 xyz 坐标。输入 216*3 的矩阵 hl1，从 i=0 开始，将每一行矩阵元素带入上述公式并循环上述操作，至 i=216 退出循环，返回值为一个 216*3 的矩阵 xl1，矩阵每一列元素依次为颜色的 x、y、z 的值。

(4) 将图像二的 hsv 值转化为 xyz 坐标。输入 200*3 的矩阵 hl2，从 i=0 开始，将每一行矩阵元素带入上述公式并循环上述操作，至 i=200 退出循环，返回值为一个 200*3 的矩阵 xl2，矩阵每一列元素依次为颜色的 x、y、z 的值。

(5) 将现有瓷砖颜色的 hsv 值转化为 xyz 坐标。输入 22*3 的矩阵 hc，从 i=0 开始，将每一行矩阵元素带入上述公式并循环上述操作，至 i=22 退出循环，返回值为一个 22*3 的矩阵 xc，矩阵每一列元素依次为颜色的 x、y、z 的值。

5.1.3 基于欧氏距离判断最接近瓷砖颜色

欧几里得度量 (euclidean metric) (也称欧氏距离) 是一种常用的距离定义，指在 m 维空间中两个点之间的真实距离，或者向量的自然长度 (即该点到原点的距离)。在我们构建的 HSV 色彩空间模型中，用欧氏距离来判断颜色接近程度。针对图像的每一种颜色，距离其欧氏距离最近的瓷砖即是人眼观感最接近的瓷砖颜色。M 维空间中两点的欧氏距离公式如下：

$$d_m(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

(1) 确定距离公式：在三维空间中，两点间的距离公式如下：

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

采用这种最常见的距离，在 HSV 色彩空间来度量我们的颜色接近程度。

(2) 具体算法如下图

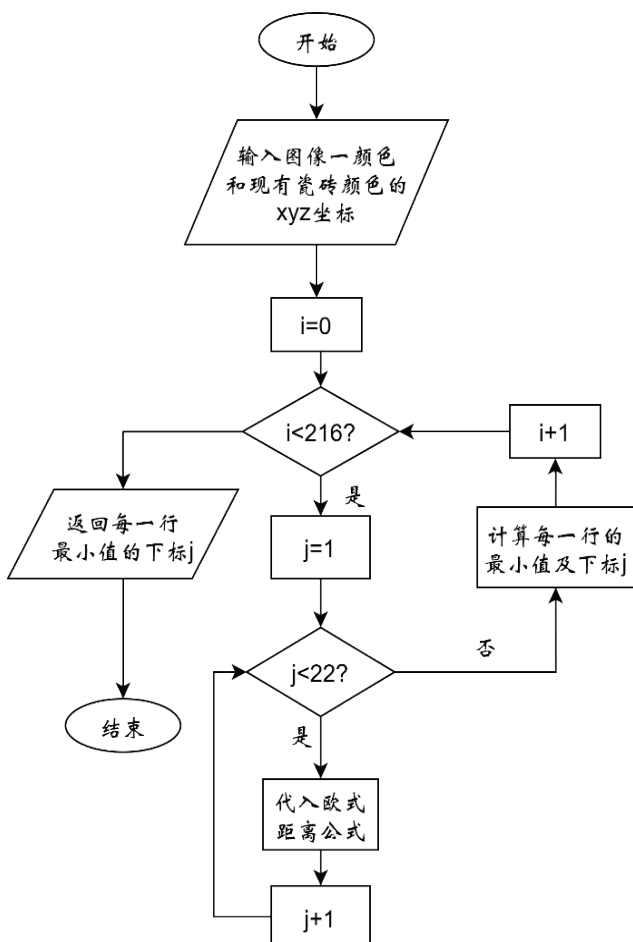


图 3 图像一欧氏距离算法流程图

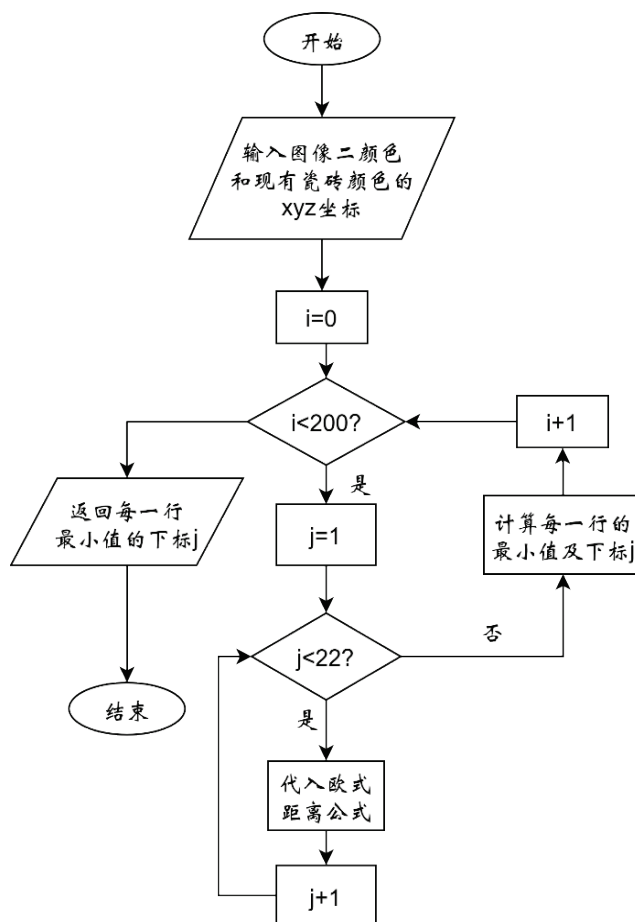


图 4 图像二欧氏距离算法流程图

图 3 为图像一与现有颜色欧氏距离计算算法流程， 图 4 为图像二与现有颜色欧氏距离计算算法流程。

5. 1. 4 建立选色模型与求解

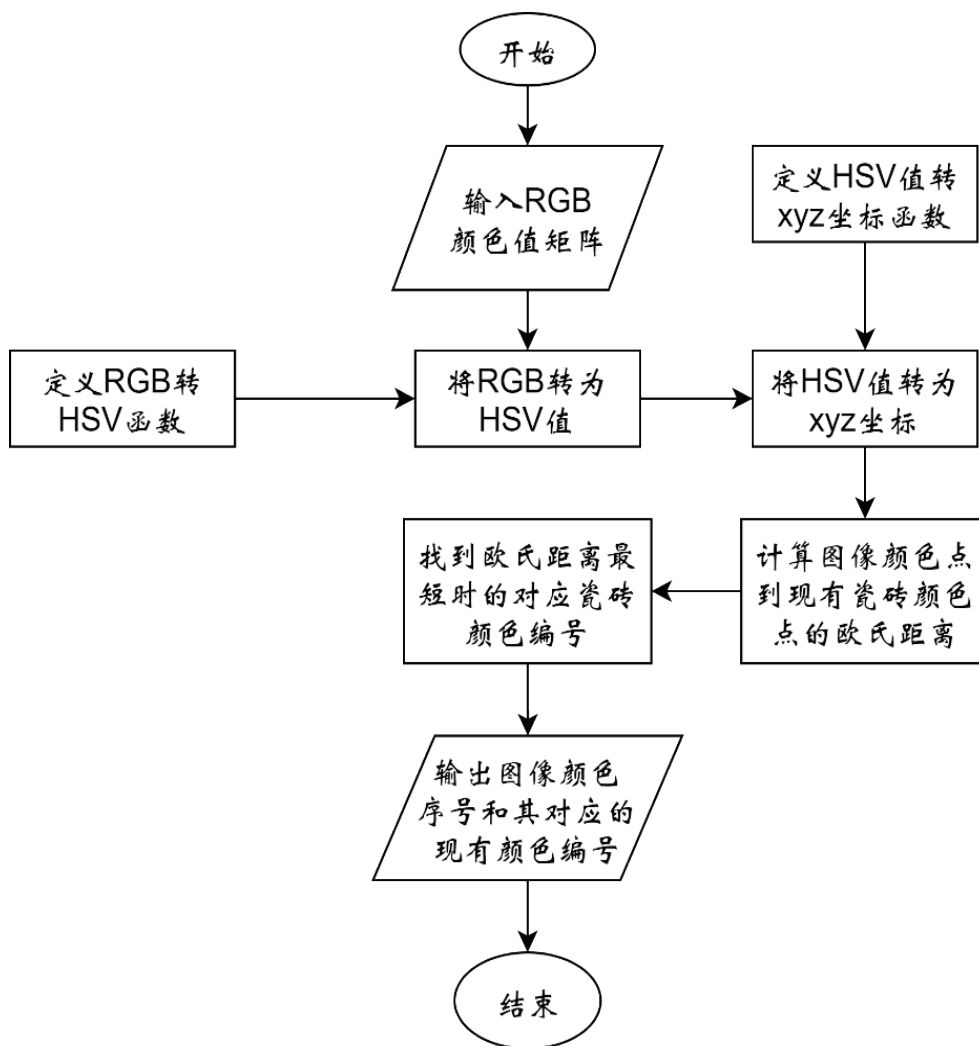


图 5 选色模型算法流程图

将上述所有模块汇总建立选色模型（如 图 5 所示），并用 python 语言编写结构化程序进行求解，并将输出结果保存在 result1.txt 和 result2.txt 文件中（代码见附录 1 和支撑文件问题一文件下的 wenti1.py 文件，最终结果见表 1 和表 2。

表 1 图像一求解结果

图像一颜色序号	瓷砖颜色编号	图像一颜色序号	瓷砖颜色编号	图像一颜色序号	瓷砖颜色编号	图像一颜色序号	瓷砖颜色编号
1	6	55	5	109	3	163	14
2	6	56	7	110	13	164	10
3	6	57	7	111	11	165	10
4	6	58	6	112	11	166	22
5	6	59	6	113	11	167	22
6	6	60	16	114	11	168	22
7	7	61	5	115	10	169	4
8	6	62	5	116	10	170	19
9	6	63	7	117	22	171	19
10	6	64	12	118	11	172	18
11	6	65	12	119	11	173	20
12	6	66	16	120	11	174	20
13	7	67	5	121	10	175	15
14	7	68	5	122	10	176	15
15	6	69	5	123	22	177	19
16	6	70	12	124	22	178	2
17	6	71	12	125	11	179	20
18	6	72	12	126	11	180	20
19	5	73	10	127	4	181	3
20	7	74	11	128	5	182	13
21	12	75	11	129	18	183	13
22	12	76	11	130	18	184	13
23	16	77	11	131	20	185	13
24	16	78	11	132	20	186	13
25	5	79	10	133	15	187	3
26	7	80	1	134	5	188	3
27	12	81	11	135	5	189	13
28	12	82	11	136	20	190	13
29	12	83	11	137	20	191	13
30	16	84	11	138	20	192	11
31	5	85	5	139	15	193	17
32	5	86	7	140	5	194	10
33	12	87	18	141	5	195	8
34	12	88	11	142	20	196	8
35	12	89	6	143	20	197	22
36	12	90	6	144	20	198	22
37	1	91	5	145	3	199	9
38	11	92	5	146	13	200	10
39	11	93	7	147	13	201	10
40	11	94	20	148	13	202	8
41	11	95	6	149	11	203	22
42	11	96	6	150	11	204	22
43	7	97	5	151	10	205	14
44	7	98	5	152	10	206	10
45	6	99	5	153	13	207	19
46	6	100	7	154	22	208	21
47	6	101	16	155	11	209	18
48	6	102	16	156	11	210	2
49	5	103	15	157	10	211	4
50	7	104	5	158	10	212	14
51	6	105	5	159	8	213	19
52	6	106	5	160	22	214	19
53	6	107	12	161	22	215	2
54	6	108	16	162	11	216	20

表 2 图像二求解结果

图像二颜色序号	瓷砖颜色编号	图像二颜色序号	瓷砖颜色编号	图像二颜色序号	瓷砖颜色编号	图像二颜色序号	瓷砖颜色编号
1	12	51	16	101	11	151	10
2	12	52	16	102	20	152	4
3	6	53	16	103	4	153	13
4	6	54	11	104	5	154	3
5	7	55	11	105	11	155	10
6	12	56	7	106	11	156	19
7	6	57	11	107	10	157	11
8	7	58	16	108	10	158	19
9	16	59	11	109	10	159	10
10	16	60	11	110	15	160	13
11	6	61	6	111	5	161	21
12	7	62	5	112	20	162	3
13	16	63	6	113	4	163	11
14	12	64	12	114	20	164	22
15	6	65	6	115	18	165	13
16	7	66	15	116	22	166	9
17	12	67	12	117	10	167	13
18	5	68	10	118	20	168	10
19	12	69	5	119	13	169	10
20	12	70	7	120	20	170	22
21	11	71	3	121	15	171	13
22	7	72	11	122	5	172	17
23	5	73	7	123	13	173	22
24	7	74	11	124	22	174	14
25	6	75	6	125	20	175	8
26	5	76	11	126	5	176	19
27	11	77	4	127	18	177	19
28	6	78	6	128	22	178	19
29	6	79	13	129	10	179	13
30	5	80	11	130	10	180	8
31	12	81	22	131	4	181	2
32	5	82	11	132	22	182	13
33	11	83	7	133	11	183	10
34	7	84	5	134	22	184	19
35	6	85	5	135	10	185	8
36	7	86	7	136	18	186	14
37	12	87	11	137	13	187	8
38	12	88	11	138	22	188	13
39	6	89	20	139	3	189	14
40	7	90	20	140	8	190	4
41	7	91	5	141	14	191	8
42	5	92	11	142	18	192	10
43	5	93	11	143	19	193	17
44	11	94	10	144	10	194	17
45	6	95	11	145	20	195	14
46	7	96	15	146	3	196	8
47	5	97	11	147	8	197	8
48	6	98	5	148	22	198	13
49	11	99	20	149	18	199	3
50	6	100	5	150	19	200	9

根据表.1、表 2 求取结果，用 matlab 输出原图像颜色及对应的瓷砖颜色对比图(matlab 代码见附录 2)，得到的最终结果如下图。

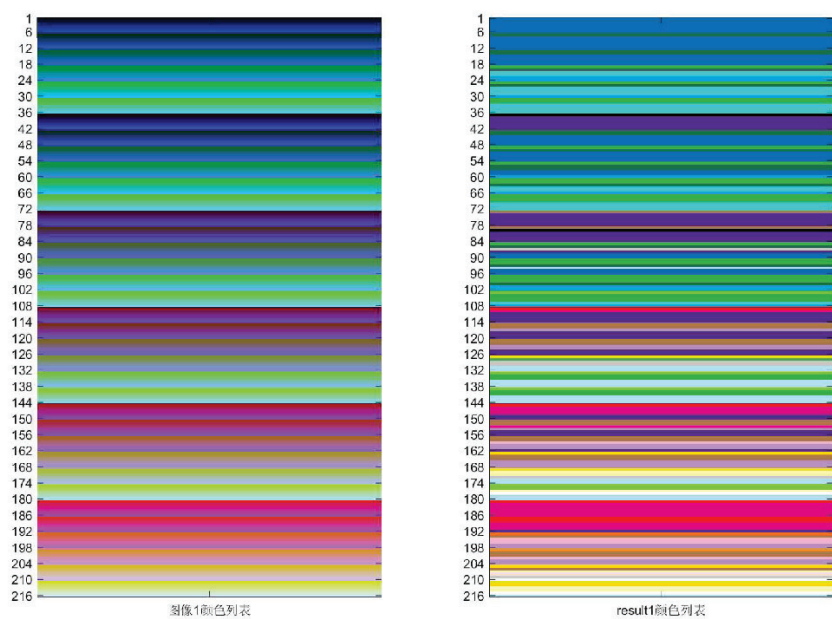


图 6 图像一与瓷砖颜色对比图

图 6 为与图像一颜色相近的瓷砖颜色，其中左边为原图像一的颜色，右边为对应的现有瓷砖的颜色，纵坐标为由大到小的原图像颜色序号。

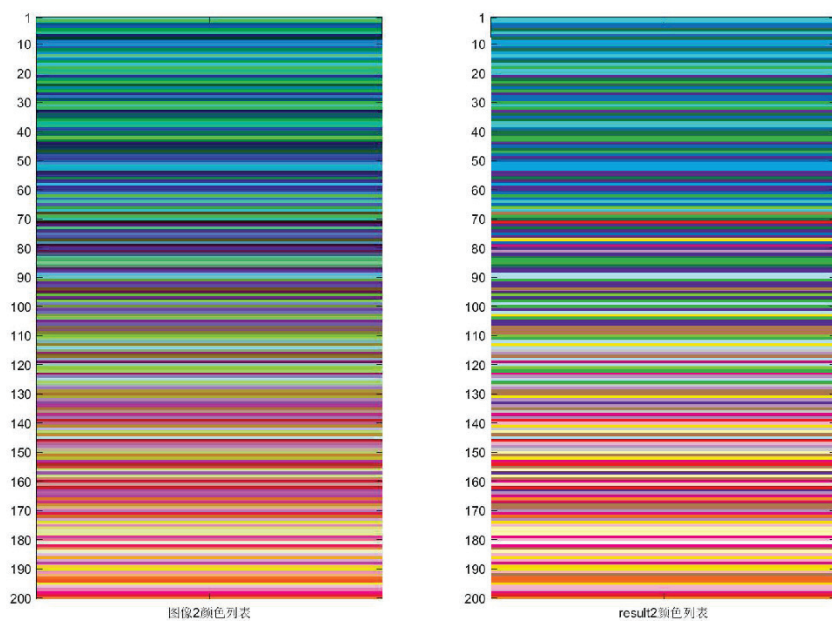


图 7 图像二与瓷砖颜色对比图

图 7 为与图像二颜色相近的瓷砖颜色，其中左边为原图像二的颜色，右边为对应的现有瓷砖的颜色，纵坐标为由大到小的原图像颜色序号。

5.2 问题二的模型建立与求解

5.2.1 拼接图像表现力打分建模

(1) 确定公式。

在 HSV 色彩空间中，H，S 代表色调分量，V 代表亮度分量。V 分量包含的是亮度信息，其受光照的影响最大，而 H 和 S 基本不受阴影或过高亮度的影响。因为我们需要研发新颜色的瓷砖来增强拼接图像的表现力，因此我们增大 H,S 的权重，降低 V 的权重。

在 HSV 空间中，我们基于欧氏距离，定义一个评价色彩接近程度的新度量值，称为 HSV 欧氏距离，其定义如下：

$$S_i = \sqrt{10 \times (x - cz(i, 1))^2 + 10 \times (y - cz(i, 2))^2 + z \times (z - cz(i, 3))^2 + 0.1 \times (\text{atan2}(y, x) - \text{atan2}(cz(i, 2), cz(i, 1)))^2}$$

$$\text{atan2}(y, x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & y \geq 0, x < 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases}$$

其中 cz 为现有瓷砖颜色 xyz 坐标的 (22*3) 矩阵， atan2 为 matlab 中的四象限反正切函数。

(2) 采用到现有瓷砖点的 HSV 欧氏距离和 S 来代表不同点的打分情况，距离和 S 越大则越优先增加该颜色。

$$S = \sum_{i=1}^{22} S_i$$

(3) 采用 matlab 编写 HSV 欧氏距离计算函数程序，并存在扩展名为 .m 文件中（详见附录 3 及支撑材料中问题 2 文件夹下的 `distan.m` 文件）。

5.2.2 非线性规划模型建立

(1) 确定目标函数。

目前没有通用算法，大多数算法都是在选定决策变量的初始值后，通过一定的搜索算法寻求最优的决策变量。我们在 HSV 空间中寻找最大距离和 S，即建立非线性规划模型，其表达式如下：

$$\max f(x) = S$$

(2) 约束条件。基于 HSV 的空间坐标系（见 图 2）得到色彩 xyz 的取值范围如下：

$$s.t. \begin{cases} x^2 + y^2 \leq 1 \\ 0 < z \leq 1 \end{cases}$$

5.2.3 最大距离求解

(1) 采用蒙特卡罗模拟 (Monte Carlo Simulation)、基于蒙特卡罗的内点法 (Interior Point Method)、遗传算法 (Genetic Algorithm)，综合求解。

(2) 发现内点法和遗传算法常常陷入局部最优解，其中基于蒙特卡罗的内点法有时陷入比蒙特卡罗结果更差的局部最优，有时可以找到更佳的局部最优，因此本模型采用蒙特卡罗模拟法求解最大距离。

蒙特卡罗模拟采用具有不确定性的变量，为 HSV 色彩空间中的位置 X、Y、Z 各分配一个随机值，在整个 HSV 色彩空间中随机取点，对每一个随机值计算到 22 个已知瓷砖在 HSV 色彩空间中的位置的 HSV 欧氏距离和。在为 X、Y、Z 变量分配许多不同的值，重复此过程，找到 HSV 欧氏距离和最大的空间点，因此可以找到接近全局最优的解。

(3) 求出第一个点后，将其加入现有瓷砖颜色中，重新求解基于 23 种已有颜色的结果，求出第二个点。按此循环往复，求出十个新的瓷砖点，分别为同时增加 1 种颜色、同时增加 2 种颜色、……、同时增加 10 种颜色的瓷砖点。

(4) 用 matlab 编写程序并保存在扩展名为.m 文件中（详见附录 4 及支撑材料中问题 2 文件夹下的 MCS.m 文件）。求取结果如表 3：

表 3 新增点的 xyz 坐标值

序号	x	y	z
1	-0.40000189159847	-0.91647965344928	0.99789546795153
2	-0.16602150806356	-0.98585859387631	0.99938008145608
3	0.10491834478536	-0.99442202074001	0.31559794926919
4	-0.99960523405402	0.00364931747123	0.28149221161571
5	0.40955429760868	-0.91223971544074	0.99737535105821
6	-0.68608148920696	0.72736717344322	0.99394239926047
7	0.61864287082599	-0.78539599085610	0.32047061230455
8	-0.99748918158357	-0.00905858329380	0.9990018877936
9	-0.31308671609526	0.94945597421590	0.99846440564304
10	0.93345515957338	-0.35838351444264	0.99685369378495

5.2.4 瓷砖颜色坐标转为 RGB 值

(1) 三维坐标转化为 HSV 值：

三维坐标可以看作笛卡尔空间直角坐标，HSV 值可类似于柱坐标，因此采用笛卡尔空间坐标转换柱坐标的方法进行转换。

由题目一中的公式倒推，转换公式如下：

$$h = \text{atan2}(y, x)$$

$$s = \sqrt{x^2 + y^2}$$

$$v = z$$

$$atan2(y, x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & y \geq 0, x < 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ undefined & y = 0, x = 0 \end{cases}$$

$$h = \begin{cases} h, & h \geq 0 \\ h + 2\pi, & h < 0 \end{cases}$$

然后将 h 由弧度制转换为角度制，其范围是 $[0, 360^\circ]$ 。

(2) HSV 数值变为 RGB 数值

将 HSV 数值按照下列公式转换为 p 、 q 、 t 分量。

$$h_i = \left\lfloor \frac{h}{60} \right\rfloor \bmod 6 \quad f = \frac{h}{60} - h_i$$

$$p = 255 \times v \times (1 - s)$$

$$q = 255 \times v \times (1 - f \times s)$$

$$t = 255 \times v \times (1 - (1 - f) \times s)$$

对于每一个颜色向量 (r, g, b) ，按照下列公式将 p 、 q 、 t 分量赋予 r, g, b 分量。

$$(r, g, b) = \begin{cases} (v, t, p) & h_i = 0 \\ (q, v, p) & h_i = 1 \\ (p, v, t) & h_i = 2 \\ (p, q, v) & h_i = 3 \\ (t, p, v) & h_i = 4 \\ (v, p, q) & h_i = 5 \end{cases}$$

(3) 用 matlab 编写程序并保存在扩展名为 .m 文件中（详见附录 5 及支撑材料中问题 2 文件夹下的 xyzttohsvtorgb.m 文件）求取结果，得到增加一种颜色、同时增加两种颜色... 同时增加十种颜色时应该增加的颜色对应的 RGB 数值。

最后列出不同的颜色开发方案，如表 4 所示。

表 4 瓷砖颜色优先开发方案表

开发方案	颜色 RGB 值	开发方案	颜色 RGB 值
同时增加 1 种颜色	27, 0, 254		
同时增加 2 种颜色	27, 0, 254 87, 0, 255		27, 0, 254 87, 0, 255
同时增加 3 种颜色	27, 0, 254 87, 0, 255 48, 0, 80	同时增加 8 种颜色	48, 0, 80 0, 72, 72 230, 0, 254 0, 253, 56 82, 0, 71 1, 253, 255
同时增加 4 种颜色	27, 0, 254 87, 0, 255 48, 0, 80 0, 72, 72		
同时增加 5 种颜色	27, 0, 254 87, 0, 255 48, 0, 80 0, 72, 72 230, 0, 254	同时增加 9 种颜色	27, 0, 254 87, 0, 255 48, 0, 80 0, 72, 72 230, 0, 254 0, 253, 56 82, 0, 71 1, 253, 255 50, 255, 0
同时增加 6 种颜色	27, 0, 254 87, 0, 255 48, 0, 80 0, 72, 72 230, 0, 254 0, 253, 56		
同时增加 7 种颜色	27, 0, 254 87, 0, 255 48, 0, 80 0, 72, 72 230, 0, 254 0, 253, 56 82, 0, 71	同时增加 10 种颜色	27, 0, 254 87, 0, 255 48, 0, 80 0, 72, 72 230, 0, 254 0, 253, 56 82, 0, 71 1, 253, 255 50, 255, 0 254, 0, 89

5.3 问题三的模型建立与求解

5.3.1 评价表现效果的新度量定义

由于题目没有提供成本的具体考量，所以我们首先定义表现效果。

根据 HSV 的特征可知，其 H, S 代表色彩信息，对表现效果的影响更大，而 V 受光照影响所以要降低其占比。因此我们定义一个新的度量，其中给平面距离高权重，立体距离小权重。这里的新度量指色彩平面立体综合距离 D，计算方式如下：

$$D = 0.9 \times D_{\text{平面}} + 0.1 \times D_{\text{立体}}$$

$$D_{\text{平面}} = \sum d_{i\text{平面}}$$

d 平面：只取两点的 X, Y 坐标，即将它们投影到一个平面上，看平面上的欧氏距离；

$$D_{\text{立体}} = \sum d_{i\text{立体}}$$

d 立体：两点在立体空间内的欧氏距离。

5.3.2 基于蒙特卡罗概率模型的建立

(1) 定义 F 函数，代表增加瓷砖 0-10 种时的符合条件率；

定义：对每一个产生的随机点，如果它能在现有的瓷砖点里找到色彩平面立体综合距离 $D < 0.3$ 的点，那么称这个点可以找到颜色最接近的瓷砖，把这种情况作为符合条件的情形，此时标志 flag 为 1，计数 N 加 1。总随机点数为 M。

$$flag = \begin{cases} 1, & D < 0.3 \\ 0, & D \geq 0.3 \end{cases}$$

$$N = \sum flag$$

$$F = \frac{N}{M}$$

用 matlab 编写主函数代码，（保存在支撑文件问题三文件夹下的 Fmain.m 文件中），代码详见附录 6

(2) 依上述公式求随机点符合条件的概率，并对每一种情况（增加瓷砖 0-10）随机计算一千万次。用 matlab 编写计算公式函数代码，（保存在支撑文件问题三文件夹下的 Fhanshugailvjisuan.m 文件中）详见附录 7。

5.3.3 评估函数 1——增长率定义

(1) 我们对得到的 11 个概率点求拟合曲线——概率为纵坐标，增加数为横坐标。求其 11 个点的斜率（即增长率），认为增长率的大小可以代表增加一块瓷砖后表现效果的提升程度。

(2) 评估函数 1：斜率函数减去平均斜率。该函数会有一个零点，这个零点就是增长率到达平均值的点，认为这一点为综合考量指标 1。

(3) 用 matlab 编写代码，（保存在支撑文件问题三文件夹下的 pingguhanshu1.m 文件中）详见附录 8。

5.3.4 评估函数 2——成本定义

(1) 由题目可知，生产每一种瓷砖的成本是相同的，但没有具体定义。我们可以结合 F 函数的值设定一个常数值 cost。对每生产一块瓷砖进行惩罚。

成本惩罚分量

$$cost = \frac{F(10) - F(0)}{10}$$

认为这一点为综合考量指标 2。

(2) 用 **matlab** 编写代码, (保存在支撑文件问题三文件夹下的 `pingguhanshu2.m` 文件中) 详见附录 9。

5.3.5 最佳成本效果点求解

概率模型求解结果:

- (1) 建立关于增长率的评估函数 1: $evaluate = k - cost$
- (2) 对符合条件的概率值二项式拟合, 如图 8 所示

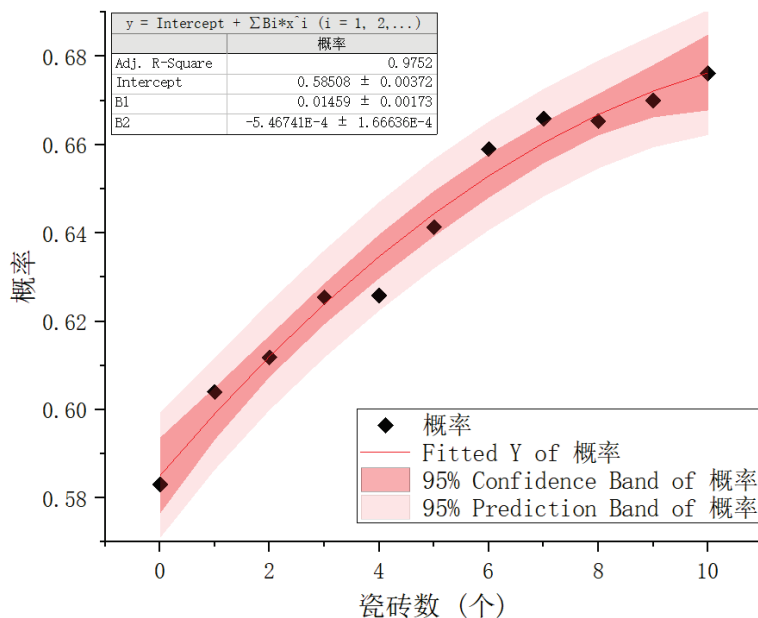


图 8 瓷砖颜色个数概率拟合图

得到一元二次方程: $y = 0.58508 + 0.01459x - 5.46741 \times 10^{-4}x^2$

其斜率方程: $k = 0.1459 - 0.00109x$

- (3) 评估函数 1 方程: $evaluate = 0.00601 - 0.00109x$

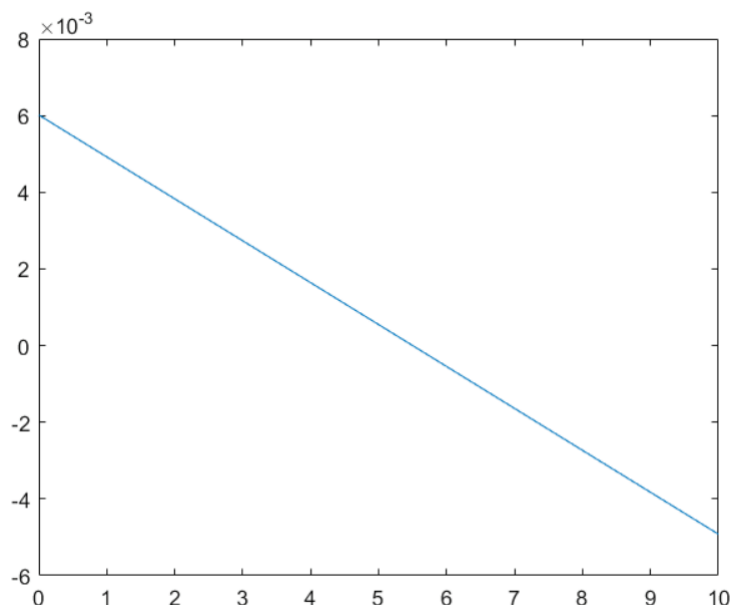


图 9 评估函数 2 函数曲线

(上述函数为确保显示效果, 都进行了保留小数点五位的操作, 实际运算中小数位更多)

得到零点 (5.5, 0)，所以得到评估指标 1 $E_1 = 5.5$ 。

(4) 建立关于成本惩罚分量的评估函数 2:

$$\text{cost} = \frac{F(10)-F(0)}{10} = \frac{0.676183-0.58309}{10} = 0.0093093$$

(5) 拟合函数减去 $F(0)$ 后，每生产一块瓷砖，减去一次成本惩罚分量 cost，找到最大值。

(6) 评估函数 2: $y_2 = 0.0052807x - 5.4671E - 4x^2$

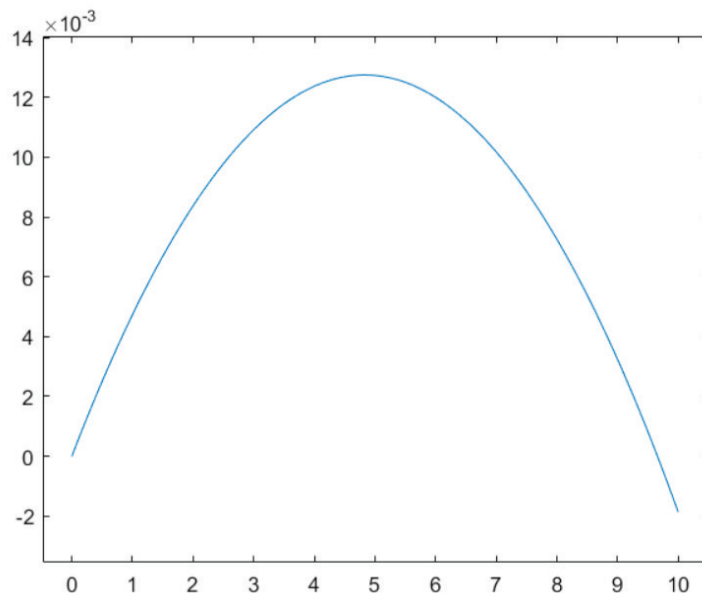


图 10 评估函数 2 函数曲线

求得其极大值点为 (4.829, 0.013)，所以得到评估指标 2 $E_2 = 4.829$ 。

5.3.6 综合指标评估

(1) 综合评估指标公式如下

$$E = 0.5E_1 + 0.5E_2$$

得到

$$E = 0.5 \times 5.5 + 0.5 \times 4.829 = 5.1645$$

根据综合评估指标，四舍五入取整后，即取 $E = 5$ 。

(2) 新增颜色的优先顺序在问题 2 中已经给出，因此这里确定在综合考虑成本和表现效果后，要新增 5 种颜色，即问题 2 中同时增加 5 种颜色时的情况。(图 11 中数字为新增颜色的 RGB 值)

表 5 最优新增颜色方案

新增颜色 1	新增颜色 2	新增颜色 3	新增颜色 4	新增颜色 5
27,0,254	87,0,255	48,0,80	0,72,72	230,0,254

六、模型的分析

6.1 问题一模型的分析

问题一模型将色彩看作为色彩空间中的离散点，不同颜色在色彩空间中的位置关系可以反映色彩之间的颜色关系，该模型通过计算空间点之间的欧氏距离，两点之间的欧氏距离越小，表示两点在空间中越接近。但是由于 RGB 颜色模型是一种计算模型颜色系统，在 RGB 色彩空间中，空间位置越接近的点，其颜色在人眼中看起来并不一定更接近；而 HSV 色彩模型是一种视觉模型系统，也就是说在 HSV 色彩空间中，空间位置越接近的点，其代表的颜色在人眼中看来也更接近。

6.2 问题二模型的分析

问题二模型将现有颜色点看作是在 HSV 色彩空间中的离散空间点。其空间分布特征可以反映选取的颜色在整个 HSV 色彩空间中的分布状况。对于离散的空间点，有三种分布特征：一是聚集分布，即颜色集中分布在 HSV 色彩空间的几个位置，聚集的分布状况不能反映出离现有颜色距离较远处的颜色，因此色彩表现力不会很高；二是均匀分布，即颜色均匀地分布在 HSV 色彩空间中，每个点之间的距离相等，在这种理想状况下，现有的颜色点可以均匀的反映出 HSV 色彩空间中每一小范围的颜色，对于色彩的表现力最强；三是随机分布，即颜色随机分布在 HSV 色彩空间中，都是可能的区域色彩分布密集，有的区域稀疏，在色彩分布密集的区域，色彩表现力强，可以更适合特定色系的颜色表现，但是对于整体色彩的表现力弱于均匀分布。因此，颜色点在 HSV 色彩空间中分布越均匀，其色彩表现力也越高。

我们定义了 HSV 欧氏距离，对 H、S 分量的坐标距离赋予高权重，对 V 分量的坐标距离赋予低权重，接下来采用蒙特卡罗方法，在 HSV 空间中随机选取空间点，计算到已有瓷砖 HSV 空间点的 HSV 欧氏距离和，找到 HSV 欧氏距离最大的空间点，再计算出其指代颜色的 RGB 分量数值，即为需要增加的颜色。

6.3 问题三模型的分析

问题三是在问题二得出结果的基础上，对同时增加 1 种、2 种、3 种……颜色作成本和表现效果的分析，即考虑到成本就是增加的种类越少越好，考虑到表现成果就是增加的种类越多越好，针对这十种情况建立目标规划模型，找出最优方案。

由于题目没有确定成本的考量标准，首先定义表现效果分量，将问题二求解的结果和 22 个已经生产出的瓷砖颜色点的集合作为现有颜色点，放入 HSV 色彩空间中；接下来使用蒙特卡罗概率模型，在色彩空间中选取 10 000 000 个随机点，计算该点到现有颜色点的 HSV 空间欧氏距离，由 HSV 的特征可知，其 H、S 代表色彩信息，对表现效果的影响更大，而 V 受光照影响所以要降低其占比，对 X、Y 坐标距离赋予 0.9 的权重，对 Z 坐标距离赋予 0.1 的权重，定义了色彩平面立体综合距离。若空间点到现有颜色点的色彩平面立体综合距离小于 0.3，认为现有颜色可以表示改颜色，因此，能表示的颜色越多，可以认为色彩的表现力越强。接下来定义成本惩罚分量，成本函数与增加颜色个数相关，每多生产一种颜色的瓷砖，就增加一次惩罚分量。将表现效果分量与成本惩罚分量叠加，得到颜色表现力与增加的颜色个数的综合指标模型。从而求解出应该增加的瓷砖颜色个数。

七、模型的评价、改进与推广

7.1 问题一模型的评价、改进和推广

该模型通过计算空间点之间的欧氏距离，两点之间的欧氏距离越小，表示两点在空间中越接近。因此，我们将颜色从 RGB 空间转换到 HSV 空间，更好地将颜色上的距离转换为了数学上的距离，并能反映出在人眼中色彩的接近距离，因此可以通过计算空间上的欧氏距离来反映颜色之间的接近程度，从而很好地代替人工对近似的颜色进行挑选。

问题一模型可以用于图像色彩的压缩。传统的方法是在 RGB 色彩空间中，将 RGB 真彩色图像转换为索引图像，但是转换后的索引图像的色彩可能会失真；如果在 HSV 色彩空间中，将 RGB 真彩色图像转换为索引图像，转换后索引图像的颜色会更加接近原图，并起到同样的压缩效果。

7.2 问题二模型的评价、改进和推广

该模型同样将色彩看作为色彩空间中的离散点，不同颜色在色彩空间中的位置关系可以反映色彩之间的颜色关系。由于色彩空间是一个连续的三维空间，而单个色彩是离散点，因此，想要用多个单个的色彩反映色彩空间中所有的颜色，是不可能的。从离散点的空间分布特征来看，均匀分布的离散颜色点最能代表空间中的所有颜色。因此采用 HSV 空间距离的最大距离和来表征离散点的空间分布状况，HSV 空间距离和越大，说明空间点分布越均匀。该模型方法将空间点的分布特征和色彩表现力相结合，定量地反映出了色彩的表现力。

由于现有的 22 个瓷砖颜色是不均匀分布的，因此 HSV 空间距离和最大，并不能说明增加型颜色后的空间点会完全均匀分布，所以模型还有待改进之处。

该模型将二维空间中离散点的分布特征引入到三维空间，实现了三维空间中离散点的部分特征的计算，算法可以推广到三维空间的 GIS 分析中，应用前景十分广泛。

7.3 问题三模型的评价、改进和推广

模型定义了色彩平面立体综合距离，通过蒙特卡罗方法，随机在 HSV 空间中选取离散的空间点，计算增加一种颜色、增加两种颜色、……、增加十种颜色后的色彩表现力，并减去生产成本分量，得到增加颜色个数的综合指标。

该模型认为所用颜色瓷砖的生产成本相同，实际上不同颜色的原料价格会存在差异，模型针对的是理想情况下，对于现实中的实际情况的求解还存在不足。

模型能够挑选出合适的瓷砖颜色进行生产，此外，还可以应用于油漆、水彩笔等对颜色种类有明显要求的生成问题中。

八、参考文献

- [1]程杰铭.色彩学[M].1 版.科学出版社, 2001.20, 23.
- [2]朱陆陆. 蒙特卡罗方法及应用[D].华中师范大学,2014
- [3]司守奎, 孙兆亮. 数学建模算法与应用[M]. 第 2 版. 北京: 国防工业出版社, 2016.
- [4]袁奋杰,周晓,丁军,吉国威,汤勇明,夏军.基于 FPGA 的 RGB 和 HSV 色空间转换算法实现[J]. 电子器件,2010,33(04):493-497.
- [5]谢君廷,王小华.一种基于 HSV 空间的颜色相似度计算方法[J].杭州电子科技大学学报,2008(01):60-63.

[6]张树波,赖剑煌.车牌定位和分割的一种综合方法[J].中山大学学报(自然科学版),2004(02):126-128+132.

九、附录

附录 1

介绍：用选色模型输出问题一求解结果的 python 代码

```
import numpy as np
import math

# 读取数据
l1 = np.loadtxt('图像1 颜色列表.txt')
l2 = np.loadtxt('图像2 颜色列表.txt')
c = np.loadtxt('现有瓷砖颜色.txt')

# 定义 rgb 转 hsv 函数, h 取值范围[0,360], s,v 取值范围[0,1]
def rgb_hsv(red, green, blue):
    global r
    r = red/255.0
    g = green/255.0
    b = blue/255.0
    ma = max(r, g, b)
    mi = min(r, g, b)

    if ma == mi:
        h = 0
    elif ma == r and g >= b:
        h = (60*((g-b)/(ma-mi))) % 360
    elif ma == r and g < b:
        h = (60*((g-b)/(ma-mi))+360) % 360
    elif ma == g:
        h = (60*((b-r)/(ma-mi))+120) % 360
    elif ma == b:
        h = (60*((r-g)/(ma-mi))+240) % 360
    else:
        h = None

    if ma == 0:
        s = 0
    else:
```

```

        s = (ma-mi)/ma

    v = ma

    return h, s, v

# 将 rbg 转换为 hsv

# 将图像 1 颜色列表转为 hsv
h11 = np.zeros(shape=(216, 3))
for i in range(0, 216):
    h11[i, 0], h11[i, 1], h11[i, 2] = rgb_hsv(l1[i, 0], l1[i, 1], l1[i, 2])

# 将图像 2 颜色列表转为 hsv
h12 = np.zeros(shape=(200, 3))
for i in range(0, 200):
    h12[i, 0], h12[i, 1], h12[i, 2] = rgb_hsv(l2[i, 0], l2[i, 1], l2[i, 2])

# 将现有瓷砖颜色转为 hsv
hc = np.zeros(shape=(22, 3))
for i in range(0, 22):
    hc[i, 0], hc[i, 1], hc[i, 2] = rgb_hsv(c[i, 0], c[i, 1], c[i, 2])

# 将 hsv 值转换为 xyz 三维坐标

# 将图像 1hsv 值转换为 xyz 三维坐标
x11 = np.zeros(shape=(216, 3))
for i in range(0, 216):
    x11[i, 0] = math.cos(h11[i, 0]*math.pi/180) * h11[i, 1]
    x11[i, 1] = math.sin(h11[i, 0]*math.pi/180) * h11[i, 1]
    x11[i, 2] = h11[i, 2]

# 将图像 2hsv 值转换为 xyz 三维坐标
x12 = np.zeros(shape=(200, 3))
for i in range(0, 200):
    x12[i, 0] = math.cos(h12[i, 0]*math.pi/180) * h12[i, 1]
    x12[i, 1] = math.sin(h12[i, 0]*math.pi/180) * h12[i, 1]
    x12[i, 2] = h12[i, 2]

```

```

# 将现有瓷砖 hsv 值转换为 xyz 三维坐标
xc = np.zeros(shape=(22, 3))
for i in range(0, 22):
    xc[i, 0] = math.cos(hc[i, 0]*math.pi/180) * hc[i, 1]
    xc[i, 1] = math.sin(hc[i, 0]*math.pi/180) * hc[i, 1]
    xc[i, 2] = hc[i, 2]

# 求欧氏距离

# 图像 1
len1 = np.zeros(shape=(216, 22))
for i in range(0, 216):
    for j in range(0, 22):
        len1[i, j] = (np.sqrt(np.sum(np.square(xc[j, :] - x11[i, :]))))
res1 = np.argmin(len1, axis=1) # 计算每行最小值的下标
result1 = np.array(res1) + 1 # 索引是以 0 开始计算, 加 1

# 图像 2
len2 = np.zeros(shape=(200, 22))
for i in range(0, 200):
    for j in range(0, 22):
        len2[i, j] = (np.sqrt(np.sum(np.square(xc[j, :] - x12[i, :]))))
res2 = np.argmin(len2, axis=1) # 计算每行最小值的下标
result2 = np.array(res2) + 1 # 索引是以 0 开始计算, 加 1

# 输出结果

# 图像 1 求解结果
k = 1
file = open('result1.txt', 'w')
file.write('序号, 瓷砖颜色编号\n')
for r in result1:
    file.write(str(k))
    file.write(',')
    file.write(str(r))
    file.write('\n')
    k += 1
file.close()

# 图像 2 求解结果
k = 1

```

```

file = open('result2.txt', 'w')
file.write('序号, 瓷砖颜色编号\n')
for r in result2:
    file.write(str(k))
    file.write(',')
    file.write(str(r))
    file.write('\n')
    k += 1
file.close()

```

附录 2

介绍：根据表 1 及表 2 中的求取结果，输出与原图像颜色对应的现有瓷砖颜色的 matlab 代码

```

%显示颜色列表
%输入m*3的RGB矩阵
%输入图像1的RGB矩阵
rgb1 = readmatrix('map1.txt');
rgbmap1 = rgb1/255;

%输入图像2的RGB矩阵
rgb2 = readmatrix('map2.txt');
rgbmap2 = rgb2/255;

%输入结果1的RGB矩阵
rgb3=readmatrix('jieguo1.txt');
rgbmap3 = rgb3/255;

%输入结果2的RGB矩阵
rgb4 = readmatrix('jieguo2.txt');
rgbmap4 = rgb4/255;

%将矩阵分为三列

%分离rgbmap1
A = mat2cell(rgbmap1,[216],[1 1 1]);
celldisp(A)

%分离rgbmap2
B = mat2cell(rgbmap2,[200],[1 1 1]);
celldisp(B)

%分离rgbmap3

```



```
C = mat2cell(rgbmap3,[216],[1 1 1]);  
celldisp(C)
```

```
%分离rgbmap4
```

```
D = mat2cell(rgbmap4,[200],[1 1 1]);  
celldisp(D)
```

```
%将矩阵转变为m*1*3的RGB矩阵
```

```
%建立图像一RGB矩阵
```

```
a = zeros(216,1,3);  
a(:,:,1) = A{1};  
a(:,:,2) = A{2};  
a(:,:,3) = A{3};
```

```
%建立图像二RGB矩阵
```

```
b = zeros(200,1,3);  
b(:,:,1) = B{1};  
b(:,:,2) = B{2};  
b(:,:,3) = B{3};
```

```
%建立结果1RGB矩阵
```

```
c = zeros(216,1,3);  
c(:,:,1) = C{1};  
c(:,:,2) = C{2};  
c(:,:,3) = C{3};
```

```
%建立结果2RGB矩阵
```

```
d = zeros(200,1,3);  
d(:,:,1) = D{1};  
d(:,:,2) = D{2};  
d(:,:,3) = D{3};
```

```
%显示颜色
```

```
%显示图像1和结果1
```

```
figure;  
subplot(1,2,1);  
image(a)  
subplot(1,2,2);  
image(c)
```

```
%显示图像2和结果2
```

```
figure;  
subplot(1,2,1);
```

```

image(b)
subplot(1,2,2);
image(d)

```

附录 3

介绍：表 4 问题二中求取 HSV 欧氏距离的函数 matlab 代码

```

%MCS中用到的HSV欧氏距离计算函数
%对于增加点后的计算为手动添加新的一行数据，修改循环，手动完成十次循环
function f = distance(x,y,z)
%数据部分
    t = [x y z];
    cz=[ 0. 0. 0.;
        0. 0. 1.;
        1. 0. 1.;
        0.53236522 0.80296639 0.96470588 ;
        -0.27610783 0.57334383 0.69019608 ;
        -0.76306767 -0.38532706 0.72941176 ;
        -0.47025333 0.28687057 0.4627451 ;
        0.23262721 -0.11202736 0.95686275 ;
        0.82788762 0.56089401 1. ;
        0.46682344 0.22856268 0.69411765 ;
        -0.06892833 -0.58623949 0.56470588 ;
        -9.50450450e-01 1.16396610e-16 8.70588235e-01 ;
        0.82697729 -0.56223532 0.89411765 ;
        0.56159958 0.67035312 1. ;
        -0.00439998 0.99999032 0.93333333 ;
        -0.88600215 -0.26498292 0.88627451 ;
        0.89969227 0.4365247 1. ;
        -4.95049505e-03 6.06260792e-19 7.92156863e-01 ;
        0.17376098 0.25173624 1. ;
        -0.25117786 -0.06842288 0.94901961 ;
        0.13301791 0.04543108 0.97647059 ;
        0.07391704 -0.22401757 0.76470588;
        -0.40000189159847 -0.91647965344928 0.99789546795153
    ];

%%计算部分
    len = 0;
    f = 0;
    for i = 1:23
        len =
sqrt(10*(x-cz(i,1))^2+10*(y-cz(i,2))^2+z*(z-cz(i,3))^2+0.1*(atan2(y,x)

```

```

-atan2(cz(i,2),cz(i,1)))^2);
        f = f + len;
    end
end

```

附录 4

介绍:问题二 5.2.3 中蒙特卡罗模拟方法求解欧式距离和最大点的坐标的 matlab 代码

```

%%蒙特卡罗模拟 Monte Carlo Simulation (MCS)
%对于循环计算十次，为手动修改运行，每一次写入'xindian.txt'一个数据，fopen取a
方式
clc
tic;
clear;
format long g
num=100000000;
x=unifrnd(-1,1,num,1);
y=unifrnd(-1,1,num,1);
z=unifrnd(0,1,num,1);
Fmax=0;
for i = 1:num
    b = [x(i), y(i), z(i)];
    if (x(i)^2+y(i)^2<=1) & (0<z(i)<=1)
        result = distance(x(i),y(i),z(i));
        if result > Fmax
            Fmax = result; %新结果大于原MAX，则更新
            B = b; %保存新MAX坐标
        end
    end
end
disp('蒙特卡罗模拟得到的色彩距离最大值为: ')
disp(Fmax)
disp('距离最大值处的x y z的坐标为: ')
disp(B)

fid = fopen('xindian.txt','a');
fprintf(fid,'%16.14f\n',Fmax);
fprintf(fid,'%16.14f\n',B);
fprintf(fid,'\n');
fclose(fid);

toc

```

附录 5
介绍：问题二 5.2.4 中将 5.2.3 中求取的 xyz 坐标值转化为 rgb 值的 matlab 代码
<pre> %该函数是将求得的十个点的xyz值转换回rgb值，并保存文件'xinzenrgb.txt' %xyz三维坐标转hsv值 xyz=[-0.40000189159847 -0.91647965344928 0.99789546795153; -0.16602150806356 -0.98585859387631 0.99938008145608; 0.10491834478536 -0.99442202074001 0.31559794926919; -0.99960523405402 0.00364931747123 0.28149221161571; 0.40955429760868 -0.91223971544074 0.99737535105821; -0.68608148920696 0.72736717344322 0.99394239926047; 0.61864287082599 -0.78539599085610 0.32047061230455; -0.99748918158357 -0.00905858329380 0.99900188779368; -0.31308671609526 0.94945597421590 0.99846440564304 0.93345515957338 -0.35838351444264 0.99685369378495]; hsv=zeros(10,3); for i = 1:10 [hsv(i,1),hsv(i,2),hsv(i,3)]=cart2pol(xyz(i,1),xyz(i,2),xyz(i,3)); if hsv(i,1)<0 hsv(i,1)=hsv(i,1)+pi*2; end hsv(i,1)=hsv(i,1)/(pi*2); end %hsv转rgb rgb=zeros(10,3); for i = 1:10 rgb(i,:)=hsv2rgb(hsv(i,:)); rgb(i,:)=round(rgb(i,:)*255); end dlmwrite('xinzenrgb.txt',rgb) </pre>

附录 6
介绍：在问题三的 5.3.2 中，基于蒙特卡罗概率模型建立的符合条件概率函数即 F 函数的 matlab 程序
<pre> %%问题3符合条件概率（F函数）主程序 dpro=zeros(11,1); </pre>

```

for c = 1:11
    dpro(c)=gl(c-1);
end
disp(dpro)

```

附录 7

介绍：在问题三 5.3.2 中，F 函数计算公式函数的 matlab 程序

```

%问题3符合条件概率（F函数）计算公式函数
function dpro = gl(n)
    cz=[ 0. 0. 0.;
        0. 0. 1.;
        1. 0. 1.;
        0.53236522 0.80296639 0.96470588 ;
        -0.27610783 0.57334383 0.69019608 ;
        -0.76306767 -0.38532706 0.72941176 ;
        -0.47025333 0.28687057 0.4627451 ;
        0.23262721 -0.11202736 0.95686275 ;
        0.82788762 0.56089401 1. ;
        0.46682344 0.22856268 0.69411765 ;
        -0.06892833 -0.58623949 0.56470588 ;
        -9.50450450e-01 1.16396610e-16 8.70588235e-01 ;
        0.82697729 -0.56223532 0.89411765 ;
        0.56159958 0.67035312 1. ;
        -0.00439998 0.99999032 0.93333333 ;
        -0.88600215 -0.26498292 0.88627451 ;
        0.89969227 0.4365247 1. ;
        -4.95049505e-03 6.06260792e-19 7.92156863e-01 ;
        0.17376098 0.25173624 1. ;
        -0.25117786 -0.06842288 0.94901961 ;
        0.13301791 0.04543108 0.97647059 ;
        0.07391704 -0.22401757 0.76470588];

    z=[ -0.40000189159847 -0.91647965344928 0.99789546795153;
        -0.16602150806356 -0.98585859387631 0.99938008145608;
        0.10491834478536 -0.99442202074001 0.31559794926919;
        -0.99960523405402 0.00364931747123 0.28149221161571;
        0.40955429760868 -0.91223971544074 0.99737535105821;
        -0.68608148920696 0.72736717344322 0.99394239926047;
        0.61864287082599 -0.78539599085610 0.32047061230455;
        -0.99748918158357 -0.00905858329380 0.99900188779368;
        -0.31308671609526 0.94945597421590 0.99846440564304;

```

```

0.93345515957338 -0.35838351444264 0.99685369378495];

jz=[cz;z(1:n,:)];

num=1000000;
x=unifrnd(-1,1,num,1);
y=unifrnd(-1,1,num,1);
z=unifrnd(0,1,num,1);
js=0;
flag=0;
zs=0;
for i = 1:num
    b = [x(i), y(i), z(i)];
    flag=0;
    if (x(i)^2+y(i)^2<=1) & (0<z(i)<=1)
        for i = 1:(22+n)
            res = 0.9*norm(b(:,1:2)-jz(i,1:2))+0.1*norm(b-jz(i,:));
            if res < 0.3
                flag = 1;
            end
        end
        if flag == 1
            js = js+1;
        end
    end
end
dpro=js/1000000;
end

```

附录 8

介绍: 在问题三 5.3.3 中,评估函数 1 的 matlab 代码

```

%问题3评估函数1
syms x
y = 0.58508+0.1459*x-5.4674E-4*x^2;
k = diff(y);
x = [1,2,3,4,5,6,7,8,9,10];
sumk = subs(k,x);
avek = mean(sumk);
evaluate = k - avek;
vpa(k);
vpa(evaluate);

```

```

x = 0:0.01:10;
tueva = 0.0060141399999999998506050591373651 -
0.0010934799999999999728372834795209*x;
tuk = 0.1459 - 0.0010934799999999999728372834795209*x;
plot(x,tueva)
x=solve(evaluate)

```

附录 9

介绍：在问题三 5.3.4 中,评估函数 2 的 matlab 代码

```

%问题3评估函数2
x = 0:0.01:10;
tuc = 0.0052807*x-5.46741E-4*x.^2;
plot(x,tuc)

syms x
df=diff(0.0052807*x-5.46741E-4*x.^2);
f=inline('0.0052807*x-5.46741E-4*x.^2');
x=solve(df);
y=f(x);
x=eval(x)
y=eval(y)

```

文件列表：

A 题附件

——查询资料

车牌定位和分割的一种综合方法_张树波.pdf

基于 FPGA 的 RGB 和 HSV 色空间转换算法实现_袁奋杰.pdf

蒙特卡洛方法及应用_朱陆陆.caj

一种基于 HSV 空间的颜色相似度计算方法_谢君廷.pdf

——代码源文件

问题 1

- duibijieguo.m
- jieguo1.txt
- jieguo2.txt
- map1.txt
- map2.txt
- result1.txt
- result2.txt
- showpoint.m
- wenti1.py
- 图像1颜色对应xyz.txt
- 图像1颜色列表.txt
- 图像2颜色对应xyz.txt
- 图像2颜色列表.txt
- 现有瓷砖颜色.txt
- 现有瓷砖颜色对应xyz.txt

问题 2

- distance.m
- MCS.m
- xindian.txt
- xinzengrgb.txt
- xyztohsvtorgb.m
- 瓷砖点数据.txt
- 新增点rgb.txt
- 新增点xyz.txt









对比的方法（最终放弃使用）
蒙特卡洛+内点法

- distance.m
- fun1.m
- ipm.txt
- MCSIPM.m
- mcsipm.txt
- nonlfun.m
- sousuo.m

遗传算法

- fun1.m
- GA.m
- ga.txt
- nonlfun.m

问题 3

-  Fhanshugailvjisuan.m
-  Fmain.m
-  pinggu1.png
-  pinggu2.png
-  pingguhanshu1.m
-  pingguhanshu2.m
-  问题3概率值二项拟合.png
-  增加瓷砖的F函数值.txt









——问题 1 结果

result1.txt






result2.txt

——中间结果与图表





问题 1

-  result11.txt
-  result22.txt
-  流程图.png
-  图像1结果对比.png
-  图像1颜色列表.txt
-  图像2结果对比.png
-  图像2颜色列表.txt
-  现有瓷砖颜色.txt

Matlab 输出原图颜色与瓷砖颜色对比图

-  duibijieguo.m
-  jieguo1.txt
-  jieguo2.txt
-  map1.txt
-  map2.txt





颜色在色彩空间中的位置

-  showpoint.m
-  图像1颜色对应xyz.txt
-  图像2颜色对应xyz.txt
-  现有瓷砖颜色对应xyz.txt

问题 2

 xindian.txt
 xinzengrgb.txt
 瓷砖点数据.txt
 新增点rgb.txt
 新增点xyz.txt

问题 3

 pinggu1.png
 pinggu2.png
 问题3概率值二项拟合.png
 增加瓷砖的F函数值.txt

A 题论文.docx