

Lab1 报告

叶茂 2200017852

1 Image Dithering

1.1 Threshold

此部分代码已给出，通过以 0.5 为阈值对图片像素实现二值化。



1.2 Uniform Random

此任务需要生成均匀的随机噪声，给图片每个像素加上噪声后进行二值化。我使用了 C++ random 库中的随机数生成器，使用 mt19937_64 作为随机数引擎，代码如下：

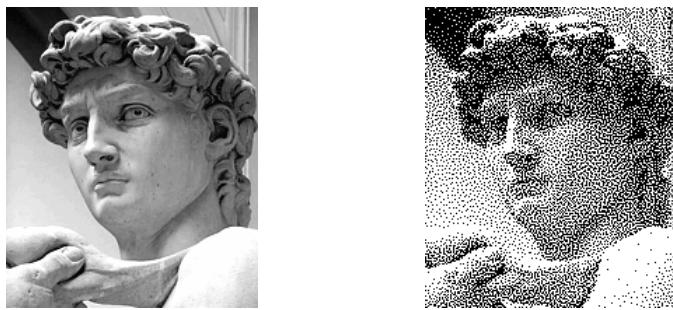
```
std::mt19937_64 eng{std::random_device{}()};
std::uniform_real_distribution<double> rd(-0.5, 0.5);
```

随后使用 `rd(eng)` 即可产生 $(-0.5, 0.5)$ 之间均匀分布的随机数。



1.3 Blue Noise Random

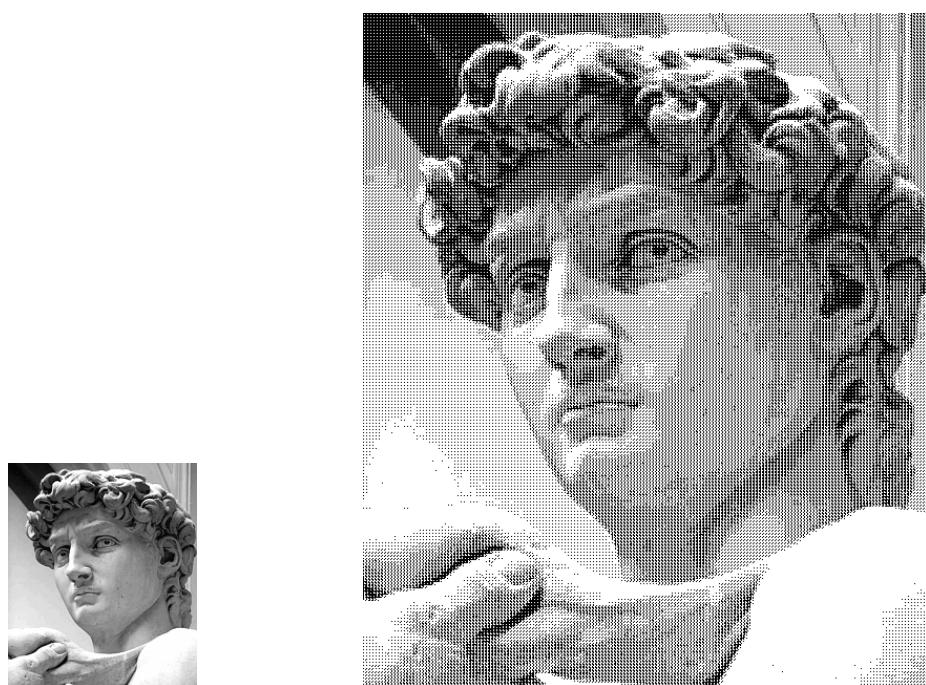
起初我简单地将原图像像素与蓝噪声图像像素相加后，以 0.5 为阈值进行二值化，发现 output 图像中大部分像素点为白色，与理想效果相去甚远。考虑到蓝噪声像素均为正值，与原图像像素相加后，所得像素值 >0.5 的概率极大，因此导致二值化后大部分像素点值为 1。因此，我将相加后的像素点的值减去 0.5 后，再进行二值化，最终实现了较好的效果。



1.4 Ordered

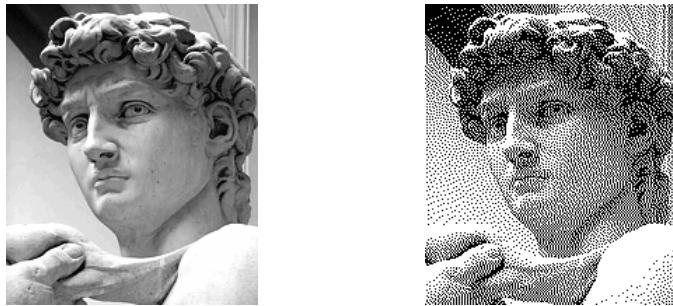
对于这个任务，我使用了讲义中给定的矩阵 $T = \begin{bmatrix} 6 & 8 & 4 \\ 1 & 0 & 3 \\ 5 & 2 & 7 \end{bmatrix}$ 。对于 input

图像中的像素，我先计算出其在 output 图像中对应的 3×3 子区域的坐标，随后将矩阵 T 的每个元素除以 9.0 后与 input 像素比较，若 input 僧素较大则得 1，否则得 0，将得到的值填入 3×3 子区域对应部分，即得到了较好的效果。



1.5 Error Diffuse

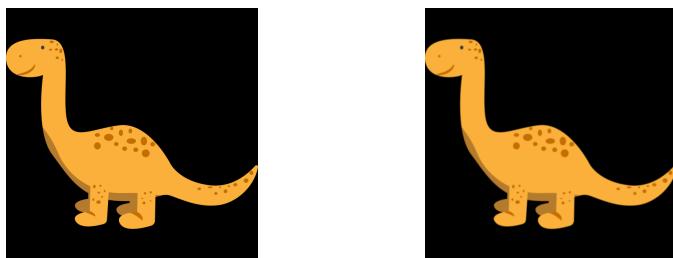
我按照讲义，计算出二值化后的像素值与原像素值的误差，并将误差加权扩散给周围像素，即完成任务。



2 Image Filtering

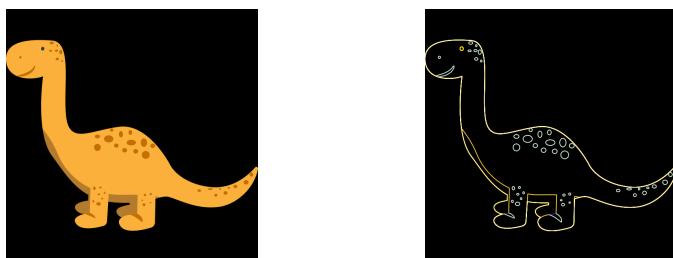
2.1 Blur

对于 input 中的每个像素，计算以此像素为中心的 $3*3$ 子区域的像素和，若有坐标超出图像边界则不计算，记录有效像素点个数 cnt，output 中对应像素值为总和 sum/cnt。



2.2 Edge Detection

类似 Blur 中的操作，分别使用竖直方向和水平方向上的Sobel卷积核，计算两个方向上的梯度。对于图像边界外的点，我在计算中将其设为 0。output 每个像素值为 input 中对应像素两个方向梯度的二范数。



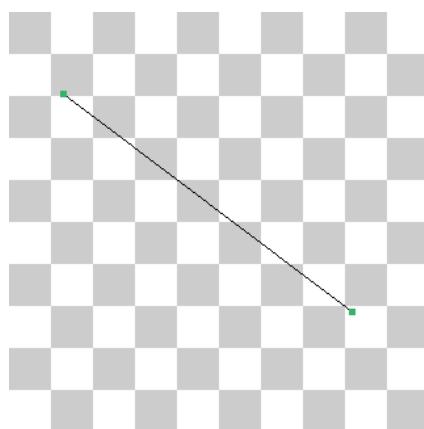
3 Image Inpainting

讲义中的 f 是需要求解的区域, f^* 是背景图, 而 g 是需要缝合的源图像。 $\nabla^2 f = \nabla^2 g$ 可以写成 $\nabla^2(f - g) = 0$ 。因此在代码中, 需要求解的 g 矩阵实际上是解与源图像的残差值。因此对于需要完美融合的边界, 自然需要初始化成背景图与源图像的差值 (offset 用于定位源图像缝合位置)。给定的代码中采用的是Gauss-Seidel迭代法, 采用已求出的 $k+1$ 次迭代解和 k 次迭代解计算当前像素点的值, 经过多次迭代可得到与真实解相近的解。



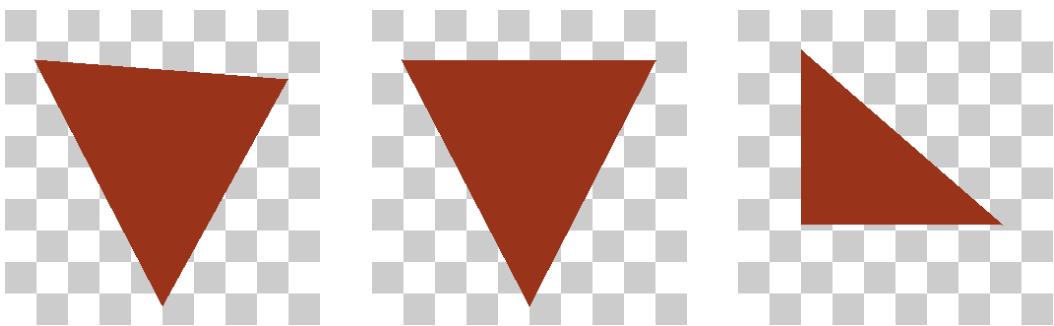
4 Line Drawing

该任务使用Bresenham算法实现。讲义中只给出了直线斜率在 $(0,1)$ 之间的情况, 若直线斜率大于 1, 则将reverse变量设置为true, 并交换每个点的 x 坐标和 y 坐标; 若起始点的 x 坐标大于终点的 x 坐标, 则交换起点和终点; 若起始点的 y 坐标大于终点的 y 坐标, 则将y_dir(表示 y 坐标是否取反) 变量设置为-1, 否则为 1, 并将两个点的 y 坐标取反。讲过上述初始化操作, 任意斜率的直线的绘制过程都转化为讲义中的情况。随后按照 Bresenham 算法初始化 error 和每次error的增量 (dy 或 delta)。需要注意的是, 绘制像素点时若reverse为true, 则须交换 x 与 y 坐标再绘制, 同时 y 坐标需要乘以y_dir的值。



5 Triangle Drawing

在这个任务中，我使用了扫描线算法。首先我先将三个点按照 y 坐标从小到大排序。随后进行特殊情况的检查：1) 若 $(y[0] - y[1]) * (x[1] - x[2]) == (y[1] - y[2]) * (x[0] - x[1])$, 则三点共线, 问题转化为绘制直线; 2) 若 $y[0] == y[1]$, 则三角形顶部水平。若三角形顶部不水平, 则初始化 x_L 和 x_R 为最高点的 x 坐标, 分别计算两条边 x 坐标的单位增量, 从小到大遍历 y 坐标, 填充每次循环中 x_L 和 x_R 之间的像素点, 并更新 x_L 和 x_R 。遍历到中间点的 y 坐标时, 若三角形底部不水平, 则需要更新其中一个增量继续遍历 y 坐标。若三角形顶部水平, 则另外计算两个增量, x_L 和 x_R 分别初始化, 类似上述过程进行绘制。



6 Image Supersampling

我将问题简化为寻找对应像素的坐标。首先计算 input 和 output 尺寸之间的比例 k_1 和 k_2 , 对于给定的超采样率 k , 求出 output 中像素点对应的 input 中的 $k_1 \times k_2$ 的矩形区域。超采样率为 k 意味着需要将 output 中的一个像素当做 $k \times k$ 个像素点看待, 而实际上 k_1, k_2 并不一定与 k 相等, 因此还需要把 input 中 $k_1 \times k_2$ 的矩形区域划分成 $k \times k$ 个矩形区域 (实际上 $k_1 = k_2$, 即为正方形), 取每个矩形区域的中心点像素进行求和, 之后取平均值, 作为 output 中该像素的值。需要注意的是, 计算比例和划分区域时, 相关计算都使用 `double` 变量, 只有在获取像素值时才将其强制转化为 `int` 类型。超采样率为 1, 3 和 5 时, 效果如下图。



(a) SSAA-1

(b) SSAA-3

(c) SSAA-5

7 Bezier Curve

这个问题按照Bezier Curve的定义不断计算得到高阶点列即可完成。起初我将 t 与 $1-t$ 和坐标的对应关系弄反了，导致绘制的曲线有误，经修正后曲线正常。

