计算机硬件基础

作业 4　BMP 图像操作

一、实验目的

BMP 图像不同颜色格式转换。256->16->灰度->黑白(抖动算法)->真彩色，等相互转换。

- 转换后的图像能以画板等软件显示。

二、设计原理

将所有输入 BMP（16 色，256 色，灰度，24bits 真彩）统一转化为 24bitsBMP 图，然后将 24bitsBMP 转化为目标格式。

在不同格式 BMP 图像之间转换的关系如下，可将所有输入 BMP 转化为 5 种输出格式中的任意一种。

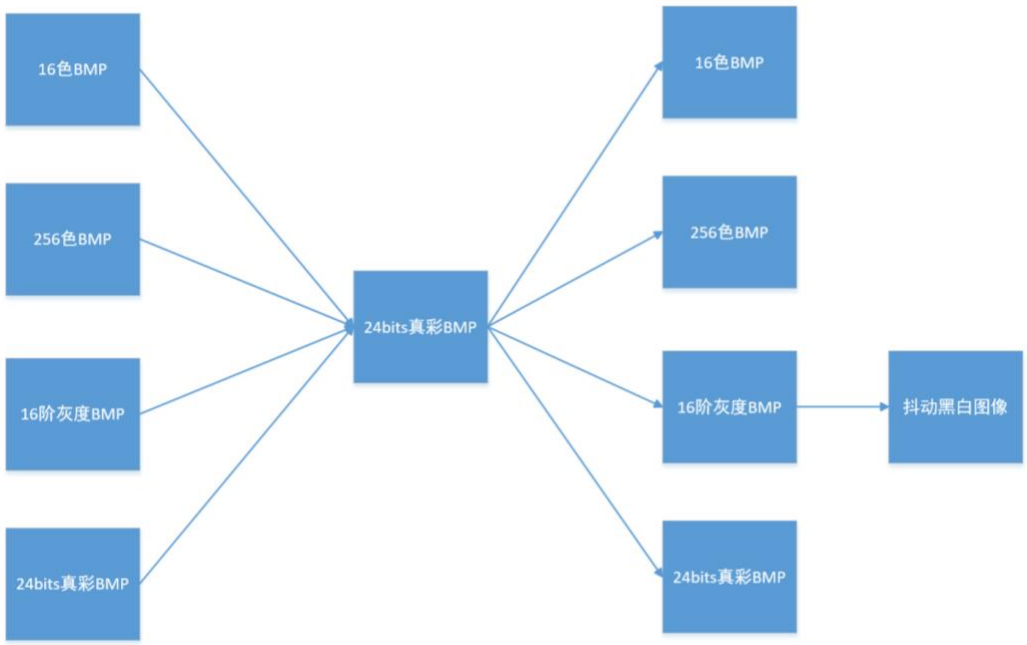


图 1 格式转换关系

转换算法建立在对 BMP 文件结构的深入理解上，BMP 文件先是 bf 和 bi 头，然后是调色板（24bits 无），最后是像素点的信息，注意每一扫描行会有的 4 字节对齐。

所以在转换过程中要先读取源文件的文件头，做修改后写入目标文件，然后写入适合目标文件的调色板，最后写入像素颜色信息或索引信息，主要在读源文件和写目标文件时的 4 字节对齐就可顺利完成一个能用画板打开的 BMP 文件。

三、程序结构

图 1 中的每个箭头都代表了一个函数（除了 24bits 真彩图到 24bits 真彩图无需函数，256 色 bmp 和 16 阶灰度到真彩色的函数为同一个函数），所以总共有 6 个转换函数，此外还有显示 BMP 文件信息的函数、获得特定 RGB 在调色板中索引的函数、灰度图调色版的初始化函数。除了函数，程序还包含预先设定的 16 色和 256 色调色板。

BMP 信息显示函数

打印文件头中的信息

```
1 void loadInfo(char *name) {
2     FILE *fp;
3     BITMAPFILEHEADER bf;
4     BITMAPINFOHEADER bi;
5     if ((fp = fopen(name, "rb")) == NULL) {
6         cout << "Open file failed." << endl;
7     }
8     fread(&bf, sizeof(bf), 1, fp);
9 
10    printf("bfType --> %c\n", bf.bfType);
11    printf("bfSize --> %d\n", bf.bfSize);
12    printf("bfOffsets --> %d\n", bf.bfOffBits);
13 
14    fread(&bi, sizeof(bi), 1, fp);
15    printf("biSize:  %d\n", bi.biSize);
16    printf("biWidth: %d\n", bi.biWidth);    //位图宽度
17    printf("biHeight:%d\n", bi.biHeight);    //位图高度
18    printf("biPlanes:%d\n", bi.biPlanes);    //颜色位面数，总为1
19    printf("biBitCount:%d\n", bi.biBitCount);    //每象数颜色位（1，2，
4，8，24）
20    printf("bmCompression:  %d\n", bi.bmCompression);
21 }
```

调色板索引函数

获得调色板中 RGB 与输入 RGB 差的绝对值和最小的色彩索引。

```
int getIndex(BYTE R, BYTE G, BYTE B, RGBQUAD *rgbquad, int size) {//得 rgb 在调色板中最接近颜色的索引
    int min_dist = 1000;
    int dist;
    int i;
    int min_index = 0;
    for (i = 0; i < size; i++) {
        dist = abs(rgbquad[i].rgbRed - R) + abs(rgbquad[i].rgbGreen - G) + abs(rgbquad[i].rgbBlue - B);
        if (dist < min_dist) {
            min_dist = dist;
            min_index = i;
        }
    }
    return min_index;
}
```

灰度调色板索引获取函数和初始化函数

索引获取函数根据输入的 RGB 运用经验公式返回对应的灰度调色板中索引。
灰度调色板初始化函数，初始化 16 阶灰度。

```
int getGrayIndex(BYTE R, BYTE G, BYTE B, RGBQUAD *rgbquad, int size) {//获得 rgb 在灰度调色板中最接近颜色的索引
    double gray = 0.299 * R + 0.587 * G + 0.114 * B;
    int index;
    index = int(gray / 17);
    return index;
}

void initGray() {
    int i;
    for (i = 0; i < 16; i++) {
```

```
        rgbquadGray[i].rgbRed = i * 17;
        rgbquadGray[i].rgbGreen = i * 17;
        rgbquadGray[i].rgbBlue = i * 17;
        rgbquadGray[i].rgbReserved = 0;
    }
}
```

转换函数

所以转换函数形式统一，两个输入分别为输入文件和结果文件的文件指针。

```
void convert4bitsTo24bits(FILE* fp, FILE* fres);
void convert8bitsTo24bits(FILE* fp, FILE* fres);
void convert24bitsTo4bits(FILE* fp, FILE* fres);
void convert24bitsTo8bits(FILE* fp, FILE* fres);
void convert24bitsToGray(FILE* fp, FILE* fres);
void convertGrayToBlackWhite(FILE* fp, FILE* fres);
```

此处以将灰度图转化为黑白抖动图的函数为例解释一下算法。
读入文件头，然后修改后将新的文件头写入新的文件。

```
fread(&bf, sizeof(bf), 1, fp);
fread(&bi, sizeof(bi), 1, fp);
bfr = bf;
bir = bi;
Height = bi.biHeight;
Width = bi.biWidth;
bfr.bfSize = 54 + 16 * 4 + Height * 4 * ((Width * 4 * 4 + 31) / 32 * 4);
bfr.bfOffBits = 54 + 64;
bir.biHeight = 4 * Height;
bir.biWidth = 4 * Width;
bir.biBitCount = 4;
fwrite(&bfr, sizeof(bfr), 1, fres);
fwrite(&bir, sizeof(bir), 1, fres);
fwrite(&rgbquadGray, sizeof(rgbquadGray), 1, fres);
```

每次读取灰度图的 1 个字节（包含两个像素点的信息），读取过程中注意四字节对齐，然后根据抖动矩阵计算对应的 4*4 像素点信息，结果储存到 BYTE BlackWhite[2000][2000]中。

```
    for (j = 0; j < Height; j++) {
```

```
        for (i = 0; i < Width; i = i + 2) {
            fread(&pixel, 1, 1, fp);
            pixel0 = pixel[0] & 0xf0 >> 4;
            pixel1 = pixel[0] & 0x0f;
            for (l = 0; l < 4; l++) {
                for (k = 0; k < 4; k++) {
                    if (pixel0 >= M[k][l]) {
                        BlackWhite[j * 4 + k][i * 4 + l] = 0xf;
                    } else {
                        BlackWhite[j * 4 + k][i * 4 + l] = 0;
                    }
                }
            }
            for (l = 0; l < 4; l++) {
                for (k = 0; k < 4; k++) {
                    if (pixel1 >= M[k][l]) {
                        BlackWhite[j * 4 + k][(i + 1) * 4 + l] = 0xf;
                    } else {
                        BlackWhite[j * 4 + k][(i + 1) * 4 + l] = 0;
                    }
                }
            }
        }
        while ((i / 2) % 4 != 0) {//注意读取时的4 字节对齐！格外注意
            fread(&pixel, 1, 1, fp);
            i = i + 2;
        }
    }
```

然后循环将结果写入目标文件，注意 4 字节对齐。

```
    for (j = 0; j < Height*4; j++) {
        for (i = 0; i < Width*4; i = i + 2) {
            index = (BlackWhite[j][i]<<4) | BlackWhite[j][i+1];
            fwrite(&index, 1, 1, fres);
        }
        while ((i / 2) % 4 != 0) {//4 字节对齐
            index = 0;
```

```
 8            fwrite(&index, 1, 1, fres);
 9            i = i + 2;
10        }
11    }
```

main 函数

读入输入输出的文件名和想要转换为的格式类型，然后自动判断输入文件的类型并调用相应函数完成操作，不涉及复杂的算法，此处不再贴代码。

四、运行实例

```
/Users/haoliu/CLionProjects/Convert/cmake-build-debug/Convert
Please input the original filename
k398a16.bmp
Please input the result filename
a.bmp
This is a 4 bits BMP picture.
Please choose the target type
----to 8 bits BMP : 0
----to 4 bits BMP : 1
----to 24 bits BMP : 2
----to Gray BMP : 3
----to BlackWhite BMP : 4
3
Convert to gray bmp success.
```

图 2 命令行交互界面实例

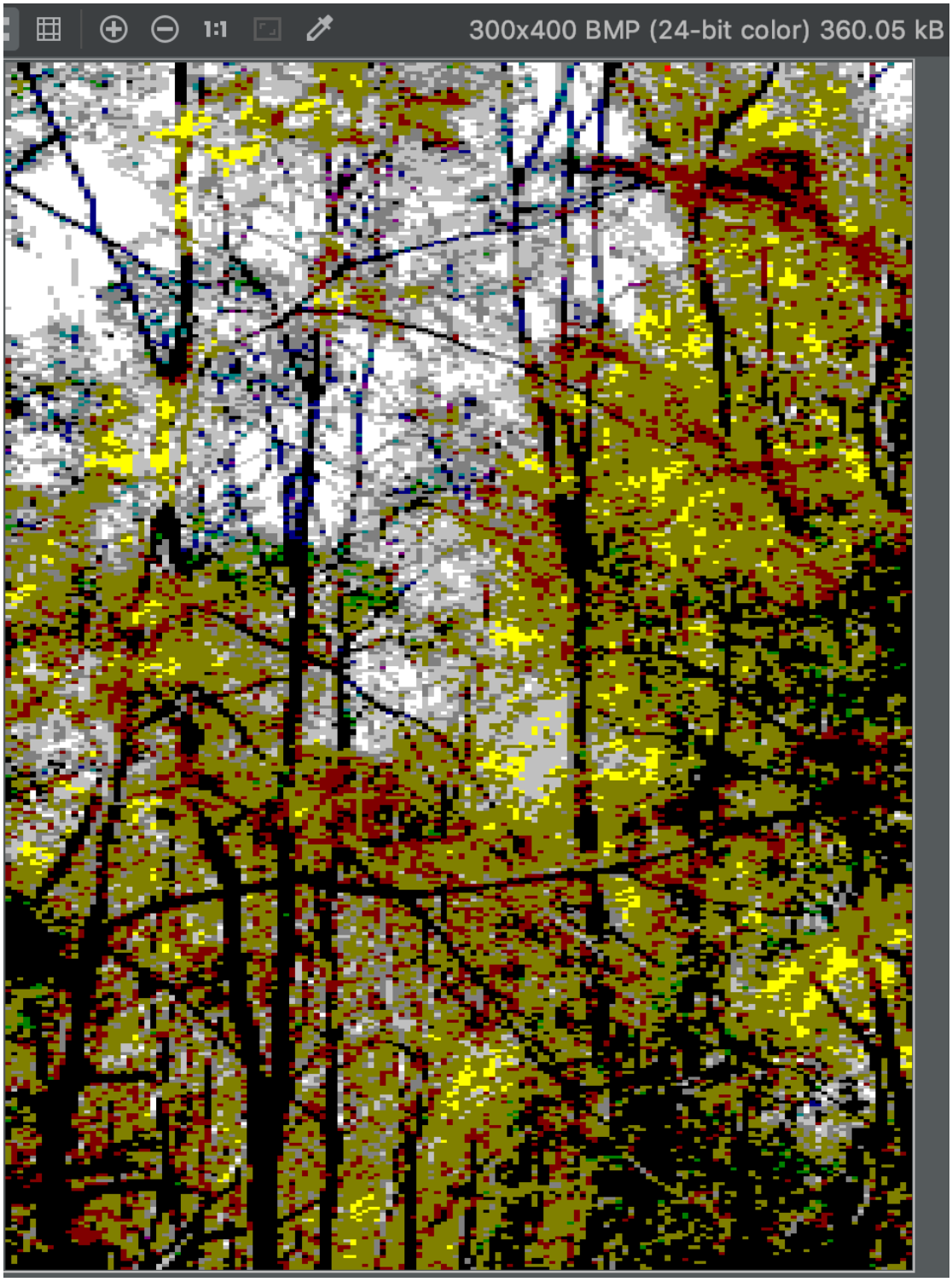
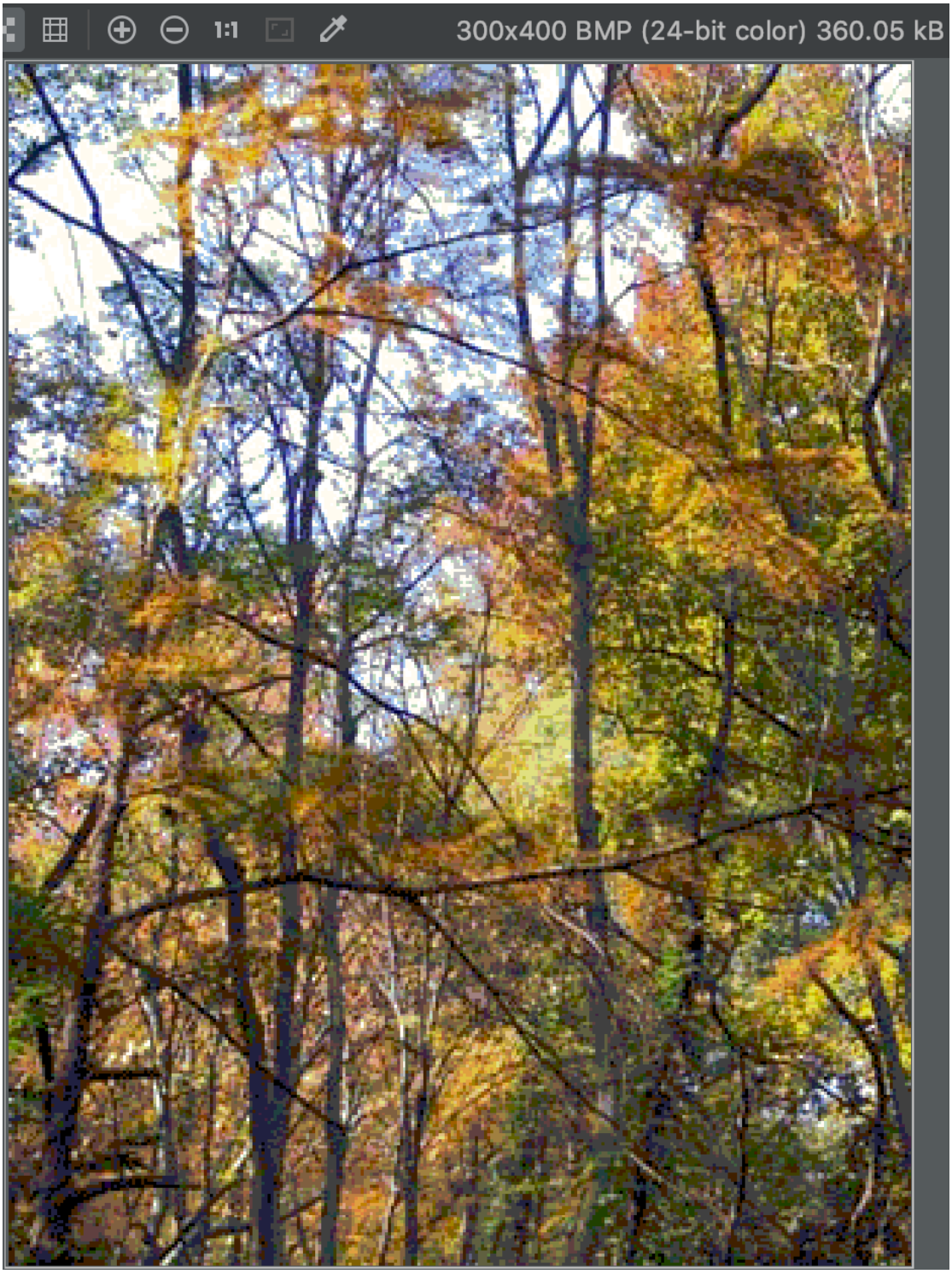


图 3 16 色转 24bits 真彩色



图表 1 256 色转换为 24bits 真彩色

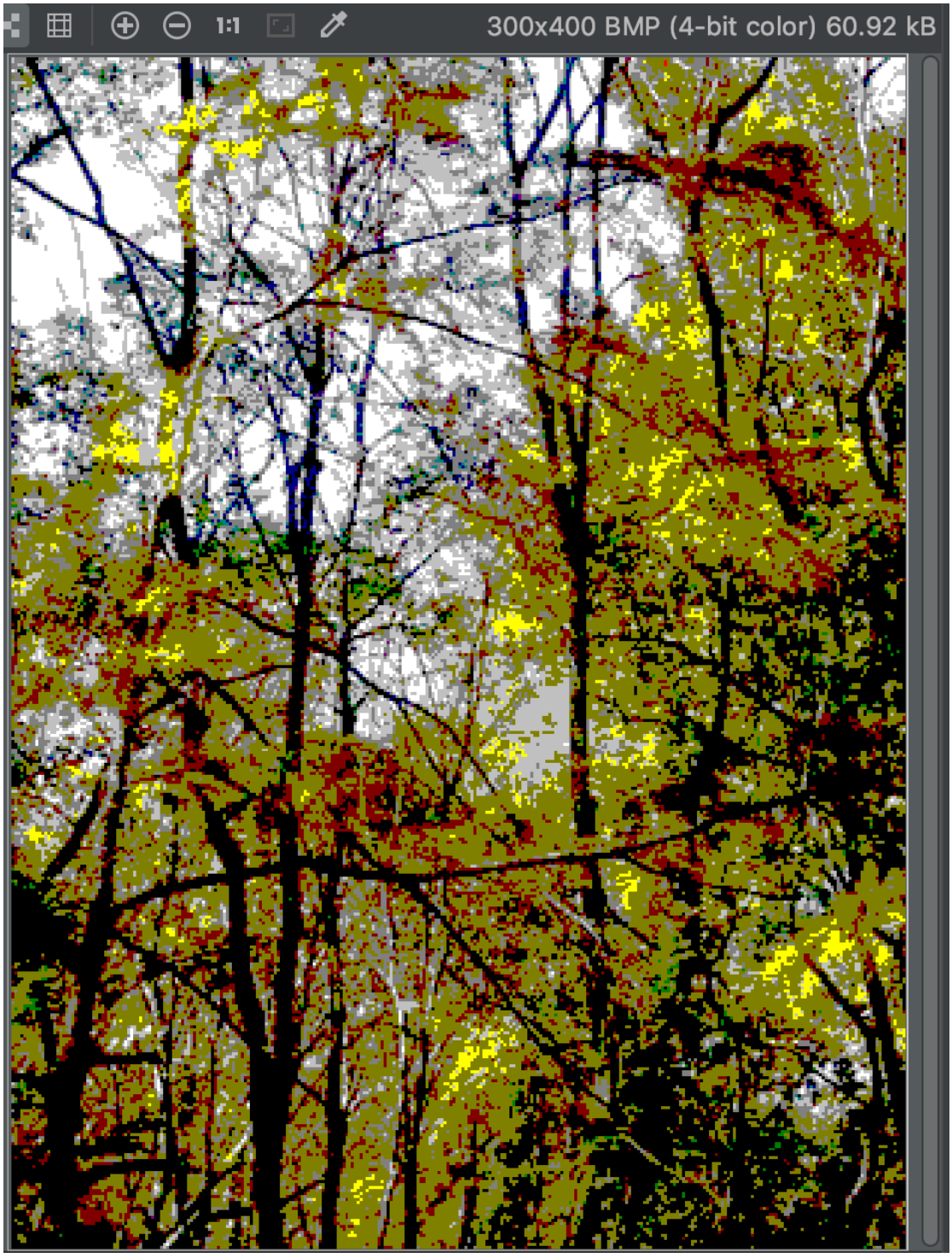


图 4 转化为 16 色图

图 5 转化为灰度图

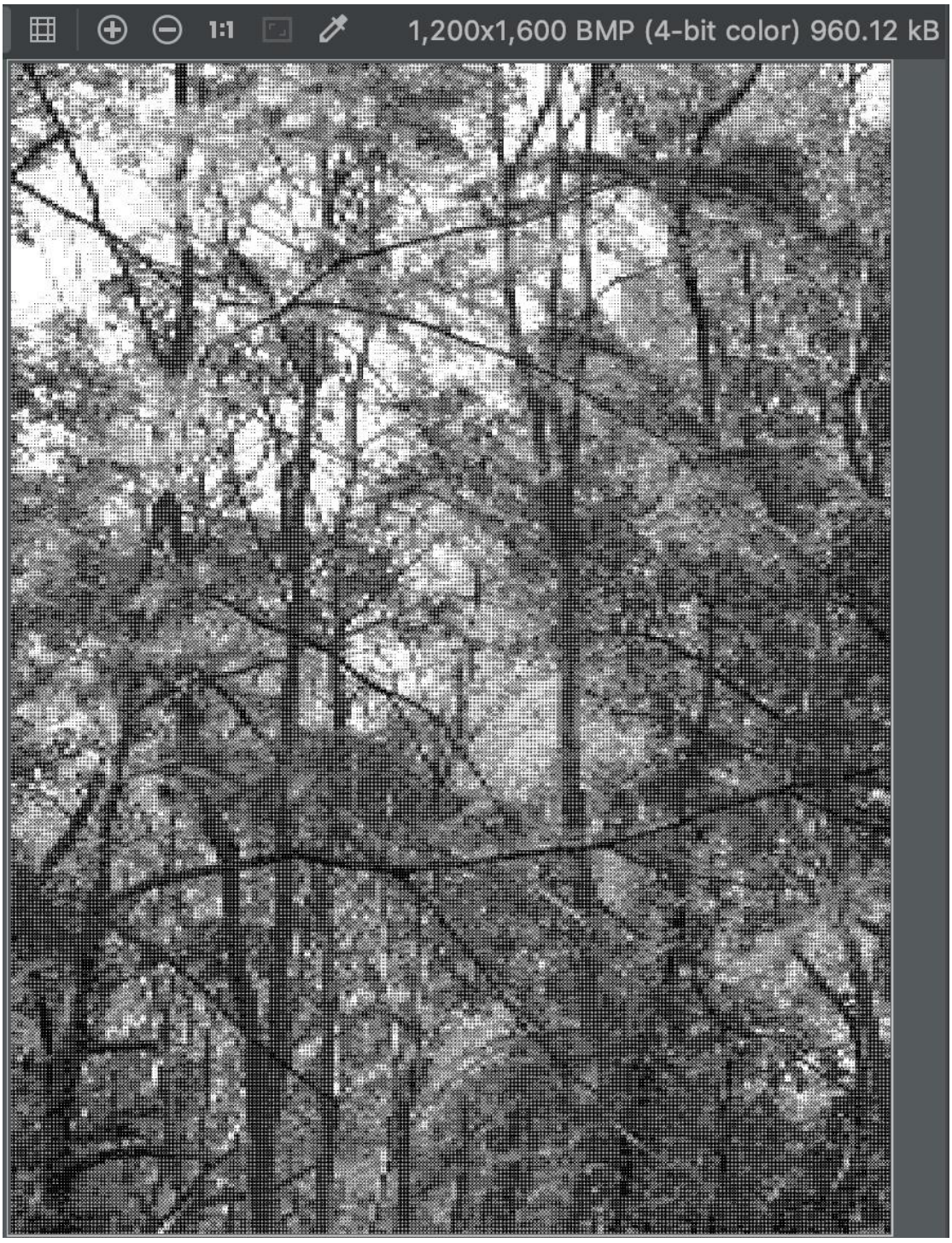


图 6 黑白抖动算法显示

图 7 黑白显示局部放大

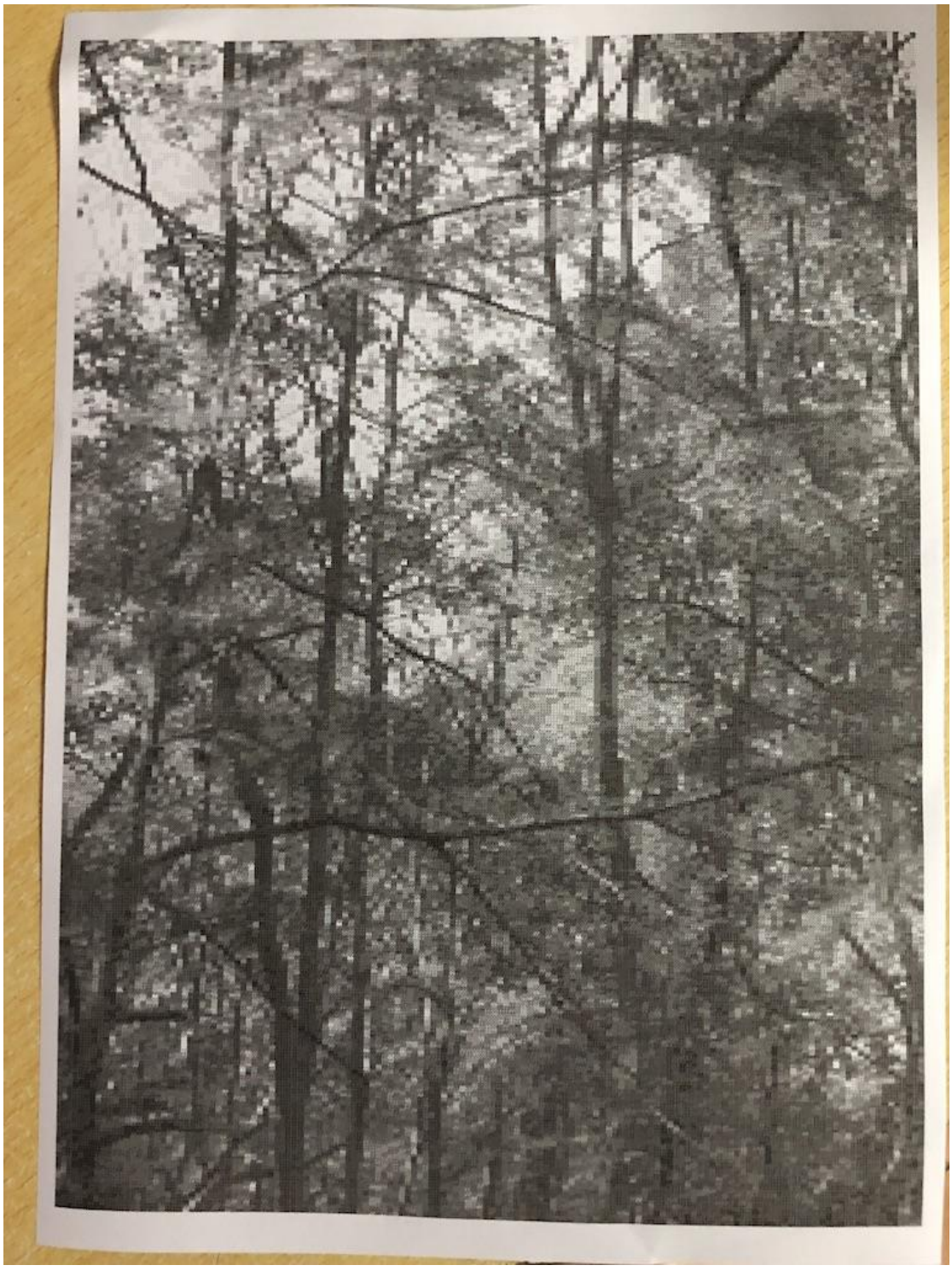


图 8 黑白图像打印效果

可以看出效果不错，远看就像是有灰度的图像，但近看会发现只有黑白两色。