機器視覺

HW 3

資工三 109590049 林敬翰

1. Convert the color image to a binary image.

Code：

```
Mat ConvertToGray(Mat img) {
    Mat gray = Mat::zeros(img.size(), CV_8UC1);
    for (int i = 0; i < img.rows; i++) {
        for (int j = 0; j < img.cols; j++) {
            Vec3b rgb = img.at<Vec3b>(i, j);
            gray.at<uchar>(i, j) = 0.3 * rgb[2] + 0.59 * rgb[1] + 0.11 * rgb[0];
        }
    }
    return gray;
}

Mat ConvertToBinary(Mat img, int threshold) {
    Mat grayimg = ConvertToGray(img);
    Mat binary = Mat::zeros(grayimg.size(), CV_8UC1);
    for (int i = 0; i < grayimg.rows; i++) {
        for (int j = 0; j < grayimg.cols; j++) {
            uchar n = grayimg.at<uchar>(i, j);
            if (n > threshold) {
                binary.at<uchar>(i, j) = 255;
            }
            else {
                binary.at<uchar>(i, j) = 0;
            }
        }
    }
    return binary;
}
```

將圖片轉成二值化沿用了 HW1 寫出來的灰階轉換、二值化轉換函式。

2. Splitting image using Quadtree

Quadtree 結構

```
struct QuadTreeNode {
```

```
    Rect rect;
    bool is_leaf;
    uchar value;
    int level=1;
    int maxLevel;
    QuadTreeNode* child[4];
};
```

我使用了 Rect 來去紀錄矩形的範圍、is_leaf 來記錄是否是最後一層、value 來記錄當前 node 的值、level 來記錄這個 node 的層數、maxLevel 來記錄這個 Quadtree 最高可以到幾層、child 來產生分支。

CreateQuadTree

```
QuadTreeNode* createQuadTree(const Mat& img, int level,int maxLevel, const Rect& rect) {
    QuadTreeNode* node = new QuadTreeNode();
    node->rect = rect;
    node->is_leaf = true;
    node->level = level;
    if (isAllPixelSame(img, rect) == 0) {
        node->value = 0;
    }
    else if (isAllPixelSame(img, rect) == 255) {
        node->value = 255;
    }
    else if(isAllPixelSame(img, rect) == 128) {
        node->value = 128;
    }
    int w = rect.width / 2;
    int h = rect.height / 2;
    if (level < maxLevel) {
        node->is_leaf = false;
        node->child[0] = createQuadTree(img, node->level + 1, maxLevel, Rect(rect.x, rect.y, w, h));
        node->child[1] = createQuadTree(img, node->level + 1, maxLevel, Rect(rect.x + w, rect.y, w, h));
        node->child[2] = createQuadTree(img, node->level + 1, maxLevel, Rect(rect.x, rect.y + h, w, h));
        node->child[3] = createQuadTree(img, node->level + 1, maxLevel, Rect(rect.x + w, rect.y + h, w, h));
    }
```

```
    return node;
}
```

在建立 Quadtree 時，我採用了遞迴的方式。先將 rect、is_leaf、level 資訊初始化，之後用 isAllPixelSame()函式去判定矩形中的值的平均值是 0、255 還是其他,如果是 0 那麼 value=0,255 那麼 value=255,其他數值的情況下 value=128。之後如果現在的 level 沒有超出 maxLevel 的話將矩形切成 4 等分並重新構築更小的 Quadtree。

isAllPixelSame

```
int isAllPixelSame(const Mat& img, const Rect& rect) {
    int sum = 0;
    for (int i = rect.y; i < rect.y + rect.height; i++) {
        for (int j = rect.x; j < rect.x + rect.width; j++) {
            sum += img.at<uchar>(i, j);
        }
    }
    int avg = sum / (rect.width * rect.height);
    if (avg == 0) {
        return 0;
    }
    else if (avg == 255) {
        return 255;
    }
    else {
        return 128;
    }
}
```

isAllPixelSame 是判定矩形中的值的平均值是 0、255 還是其他，跑過每個 pixel 後將所有的值都加起來除以長 x 寬，如果是 0 則回傳 0、255 回傳 255、其他的場合回傳 128。

drawQuadTree

```
void drawQuadTree(Mat& img, const QuadTreeNode* node) {
    if (node->value == 0) {
        rectangle(img, node->rect, Scalar(0, 0, 0), -1);
    }
    else if (node->value == 255) {
        rectangle(img, node->rect, Scalar(255, 255, 255), -1);
    }
    else {
        rectangle(img, node->rect, Scalar(128, 128, 128), -1);
    }
    if (!node->is_leaf) {
        for (int i = 0; i < 4; i++) {
            drawQuadTree(img, node->child[i]);
        }
    }
}
```

drawQuadTree 是把構建好的 Quadtree 畫出來，利用遞迴的方式去跑過每個 node，然後將其對應的 rect 把他畫出來到 img 上。

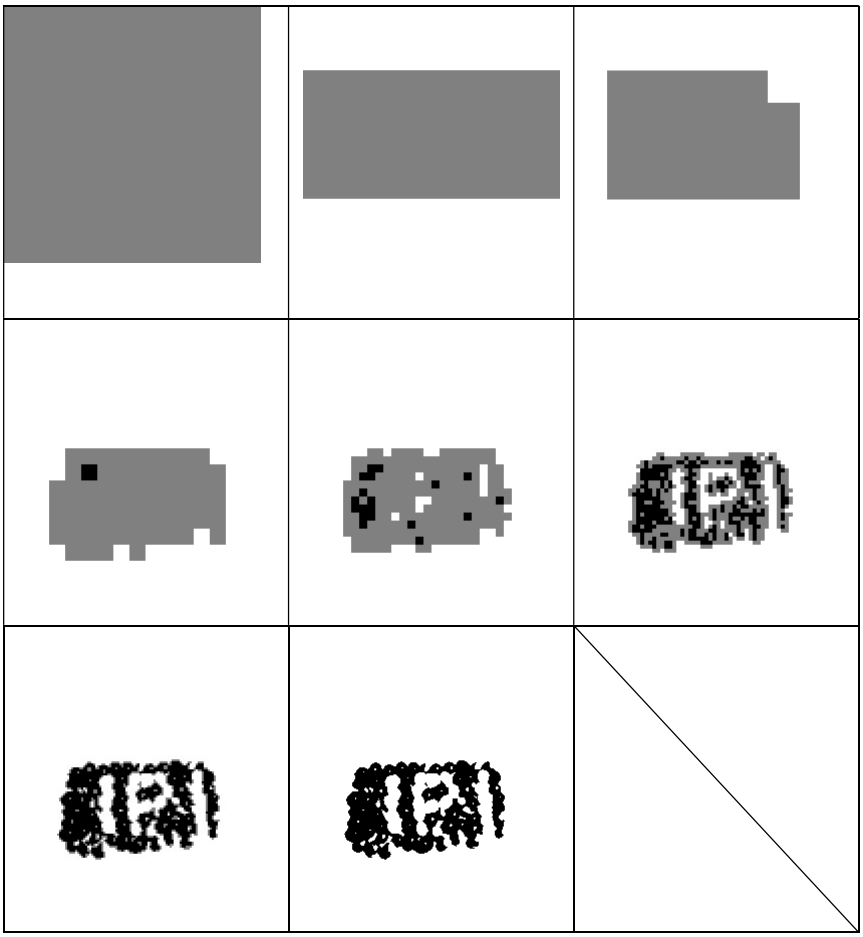
Output *用 1.jpg 當範例

```
void picl() {
    Mat im = imread("source image/1.png");
    Mat img = ConvertToBinary(im, 135);
    for (int i = 2; i <= int(log2(img.cols)) + 1; i++) {
        QuadTreeNode* root = createQuadTree(img, 1, i, Rect(0, 0, img.cols, img.rows));
        Mat img_tree(img.size(), CV_8UC3);
        drawQuadTree(img_tree, root);
        imwrite("output/picl/picl_Layer_" + to_string(i - 1) + ".png", img_tree);
        delete root;
    }
}
```

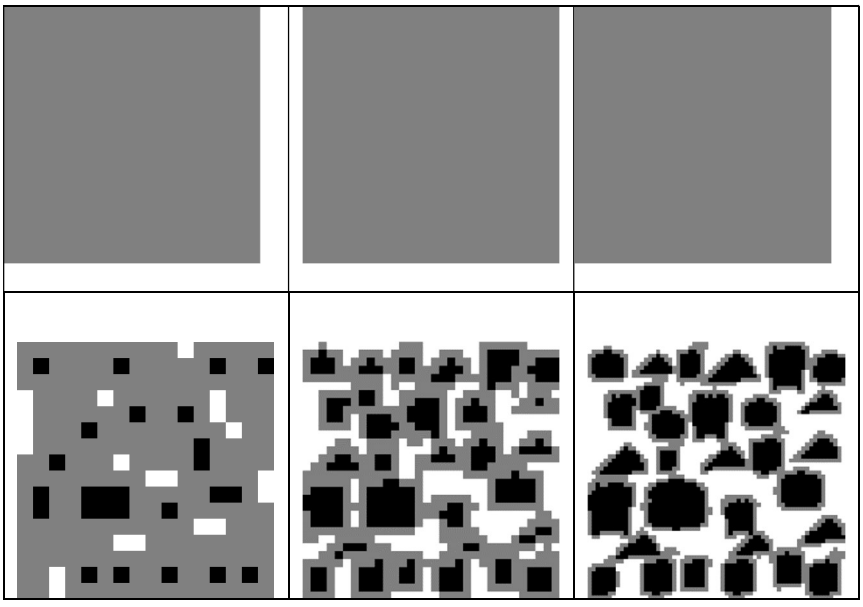
讀入 1.png 後，將其轉成 binary image，之後跑迴圈讓 maxLevel 從 2 到其 log2(長寬值)去建構 Quadtree，再用 drawQuadTree 把他畫出來並輸出。

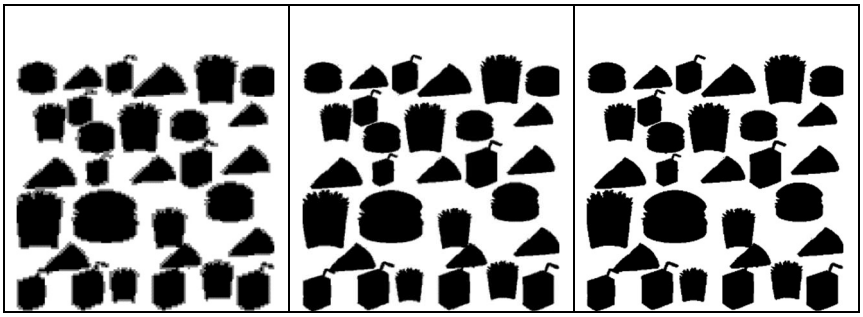
輸出

1.png

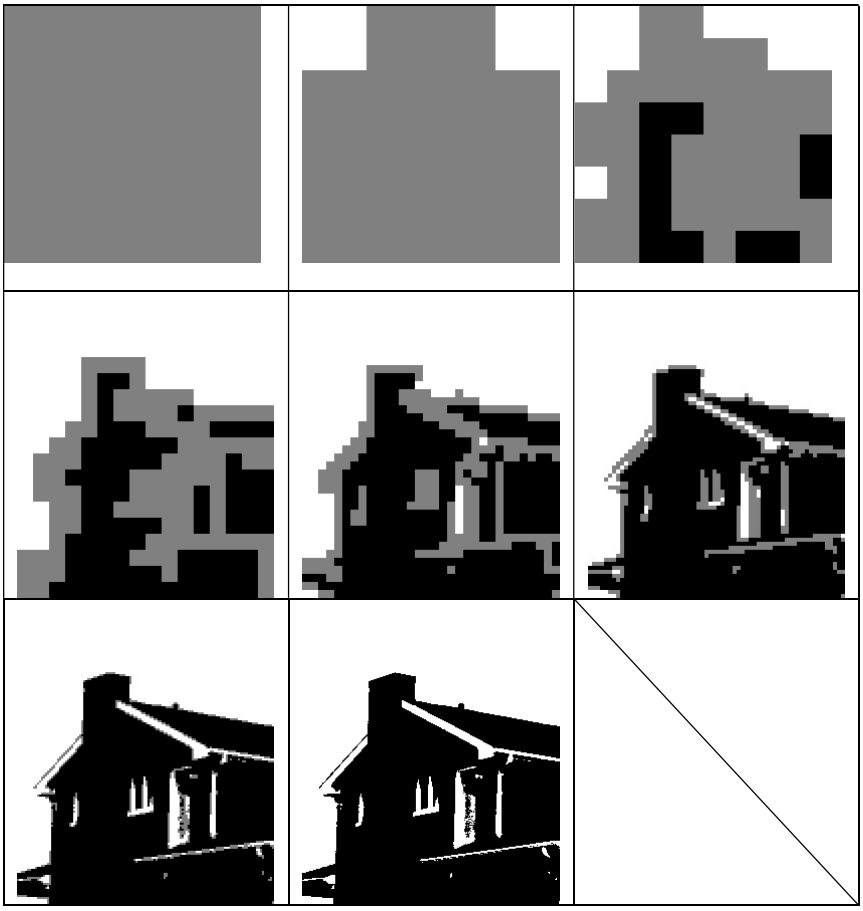


2.png





3.png



4.png