機器視覺 HW3 資工三 109590041 范遠皓

環境

Visual Studio 2019、C++ 、openCV2.4.13.6

Splitting image using Quadtree

灰階、二值化同 hw1

```cpp
// 轉灰階
Mat ConvertToGray(const Mat& colorImage) {
    Mat grayImage(colorImage.size(), CV_8UC3);

    for (int i = 0; i < colorImage.rows; i++) {
        for (int j = 0; j < colorImage.cols; j++) {
            Vec3b pixel = colorImage.at<Vec3b>(i, j);
            int grayValue = 0.3 * pixel[2] + 0.59 * pixel[1] + 0.11 * pixel[0];
            grayImage.at<uchar>(i, j) = grayValue;
        }
    }
    return grayImage;
}

// 灰階二值化
Mat ConvertToBinary(const Mat& colorImage, uchar threshold = 128) {
    Mat grayImage = this->ConvertToGray(colorImage);
    Mat binaryImage(grayImage.size(), CV_8UC1);

    for (int i = 0; i < grayImage.rows; i++)
        for (int j = 0; j < grayImage.cols; j++)
            binaryImage.at<uchar>(i, j) = grayImage.at<uchar>(i, j) > threshold ? 255 : 0;
    return binaryImage;
}
```

Threshold、layer 根據文件設定

```cpp
class ImageInfo
{
public:
    string _fileName;
    int _threshold;
    int _layer;

    ImageInfo(string fileName, int threshold, int layer) {
        this->_fileName = fileName;
        this->_threshold = threshold;
        this->_layer = layer;
    };
};

int main() {
    const int IMAGE_NUM = 4; // 圖片數
    const string IMAGE_FOLDER = "..\\image"; // 圖片存放資料夾
    const string IMAGE_PATH_FORMAT = IMAGE_FOLDER + "\\%s.png";
    vector<ImageInfo> images;

    // 圖片檔名、threshold、layer 設定
    images.push_back(ImageInfo("1", 135, 8));
    images.push_back(ImageInfo("2", 245, 9));
    images.push_back(ImageInfo("3", 155, 8));
    images.push_back(ImageInfo("4", 254, 9));
```

將圖片二值化，因二值化時把通道換成 1 ，但實作 Quadtree 時顏色用通道 3 所以先將二值化圖片變成通道 3 顏色不變。接著建立 Quadtree 分裂節點，之後依據 layer 繪製成圖片表示。

```
// 讀取圖片、二值化
Mat colorImage = imread(IMAGE_PATH);
Mat binaryImage = library.ConvertToBinary(colorImage, image._threshold);

imshow("true-color " + IMAGE_PATH, colorImage);
imshow("binary " + IMAGE_PATH, binaryImage);
imwrite(format(IMAGE_PATH_FORMAT.c_str(), (IMAGE_NAME + "_binary").c_str()), binaryImage);

// 將 binaryImage 轉為 3 通道
Mat binaryImageChannel3 = binaryImage;
binaryImageChannel3 = Mat(binaryImage.size(), CV_8UC3);
for (int i = 0; i < binaryImage.rows; i++)
    for (int j = 0; j < binaryImage.cols; j++)
        binaryImageChannel3.at<Vec3b>(i, j) = Vec3b(binaryImage.at<uchar>(i, j), binaryImage.at<uchar>(i, j), binaryImage.at<uchar>(i, j));

// 建立 Quadtree 並進行切分
QuadtreeNode root = QuadtreeNode(Rect(0, 0, binaryImage.cols, binaryImage.rows), 0);
root.SplitNode(binaryImageChannel3);

// 根據 layer 繪製 Quadtree 圖片
for (int layer = 1; layer <= image._layer; layer++)
{
    Mat resultImage(binaryImage.size(), CV_8UC3);
    root.DrawNode(resultImage, layer);
    imshow("splitted layer" + to_string(layer) + IMAGE_PATH, resultImage);
    imwrite(format(IMAGE_PATH_FORMAT.c_str(), (IMAGE_NAME + "_splitted layer" + to_string(layer)).c_str()), resultImage);
}
```

功能與上面相同，可傳入 colorImage，但每次都要重新切分速度較慢所以改成上方寫法。

```
// Quadtree
Mat SplitImageByQuadtree(const Mat& srcImage, int layer = INT_MAX) {
    Mat splitImage = srcImage;
    Mat resultImage(srcImage.size(), CV_8UC3);

    // 將 binaryImage 轉為 3 通道
    if (srcImage.type() != CV_8UC3)
    {
        splitImage = Mat(srcImage.size(), CV_8UC3);
        for (int i = 0; i < srcImage.rows; i++)
            for (int j = 0; j < srcImage.cols; j++)
                splitImage.at<Vec3b>(i, j) = Vec3b(srcImage.at<uchar>(i, j), srcImage.at<uchar>(i, j), srcImage.at<uchar>(i, j));
    }

    // 切分並繪製 Quadtree 圖片
    QuadtreeNode root = QuadtreeNode(Rect(0, 0, srcImage.cols, srcImage.rows), 0);
    root.SplitNode(splitImage, layer);
    root.DrawNode(resultImage);
    return resultImage;
}
```

QuadtreeNode:: SplitNode(const Mat& image, int maxLevel = INT_MAX)

根據 Quadtree 對 image 進行切分節點，多於一種顏色則繼續分裂，只有一種顏色則設定該顏色為節點的顏色並且不繼續分裂。

```
// 分裂節點
bool SplitNode(const Mat& image, int maxLevel = INT_MAX) {
    // 不是葉節點或達到最大上限不分裂
    if (_level > maxLevel || !_isLeaf)
        return false;

    bool continueSplit = false;

    // 多於一種顏色繼續分裂
    if (CheakColor(image)) {
        continueSplit = true;
        _isLeaf = false;

        int halfWidth = _rect.width / 2;
        int halfHeight = _rect.height / 2;

        _childrens[0] = new QuadtreeNode(Rect(_rect.x, _rect.y, halfWidth, halfHeight), _level + 1);
        _childrens[1] = new QuadtreeNode(Rect(_rect.x + halfWidth, _rect.y, halfWidth, halfHeight), _level + 1);
        _childrens[2] = new QuadtreeNode(Rect(_rect.x, _rect.y + halfHeight, halfWidth, halfHeight), _level + 1);
        _childrens[3] = new QuadtreeNode(Rect(_rect.x + halfWidth, _rect.y + halfHeight, halfWidth, halfHeight), _level + 1);

        for (int i = 0; i < 4; i++)
            _childrens[i]->SplitNode(image, maxLevel);
    }
    else // 只有一種顏色
        _color = image.at<Vec3b>(_rect.x, _rect.y);

    return continueSplit;
}
```

QuadtreeNode:: DrawNode(Mat& image, int maxLevel = INT_MAX)

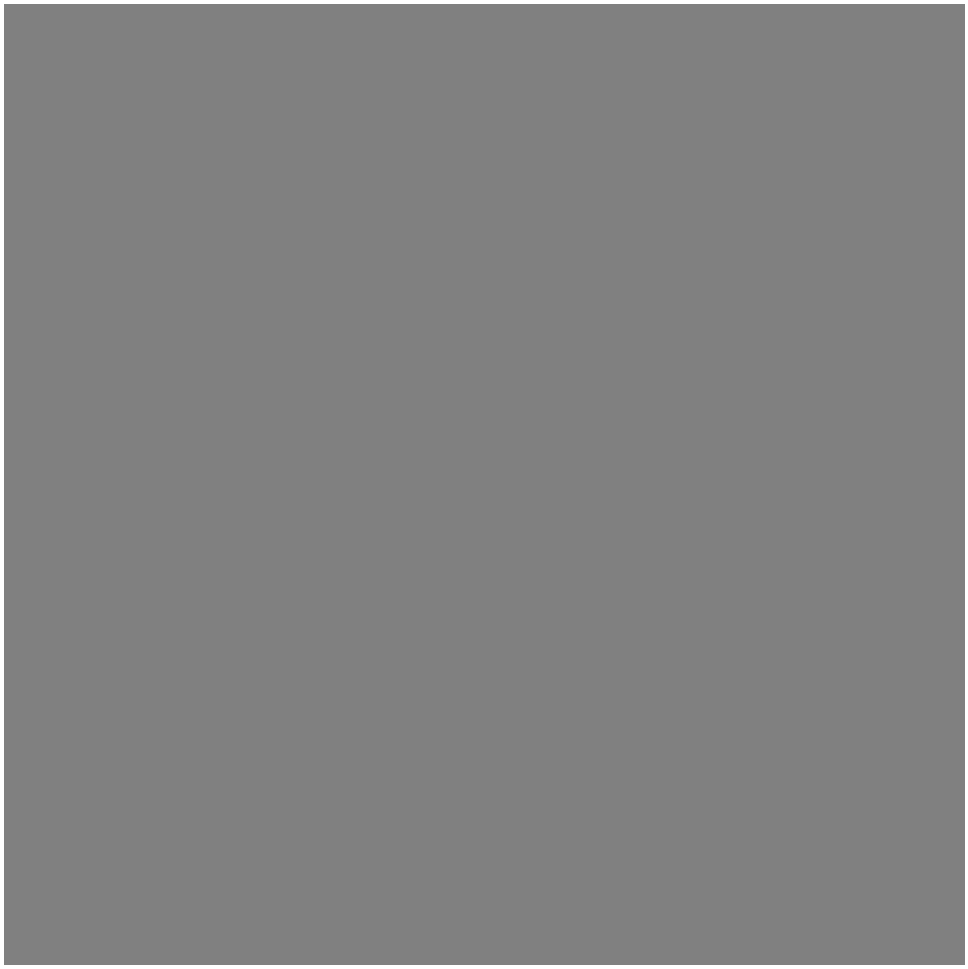
根據 layer 將節點繪製到圖片上。

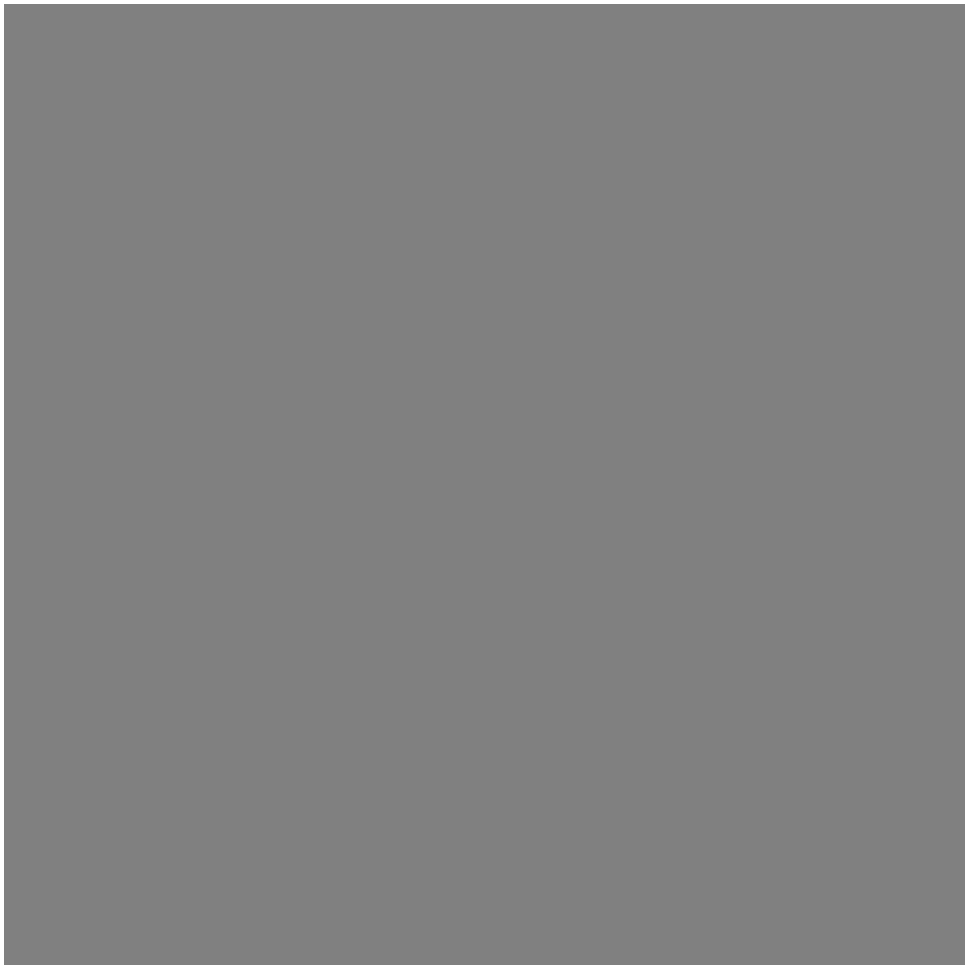
```
// 將 node 繪製到 image
void DrawNode(Mat& image, int maxLevel = INT_MAX) {
    if (this->_isLeaf || this->_level >= maxLevel) {
        for (int i = _rect.x; i < _rect.x + _rect.width; i++)
            for (int j = _rect.y; j < _rect.y + _rect.height; j++)
                image.at<Vec3b>(i, j) = this->_color;
    }
    else {
        for (int i = 0; i < 4; i++)
            this->_childrens[i]->DrawNode(image, maxLevel);
    }
}
```

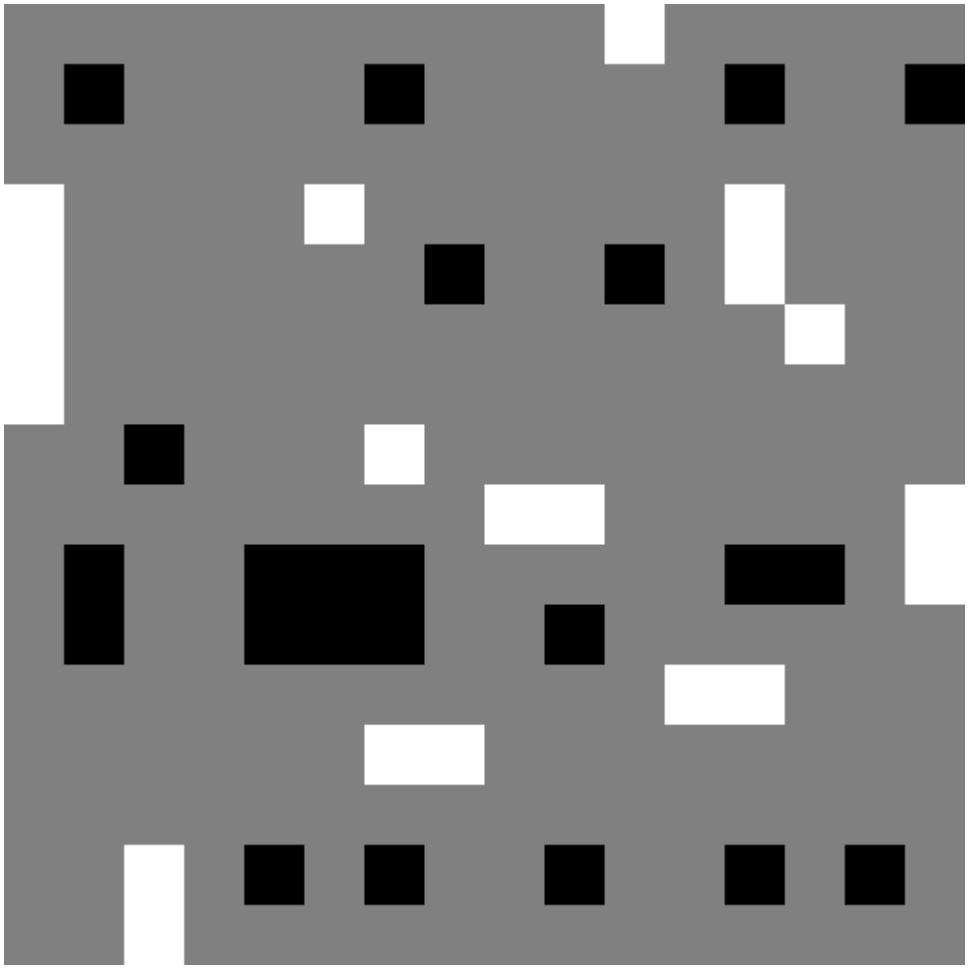
結果圖片

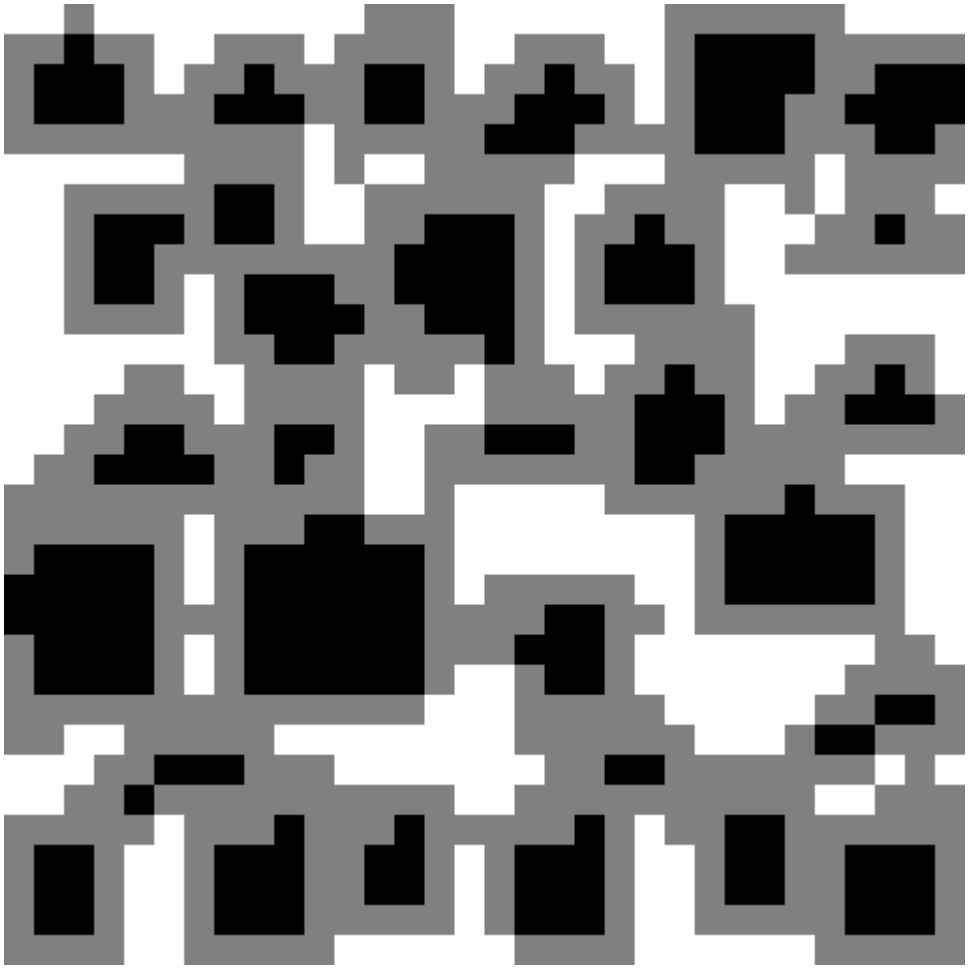
1.png

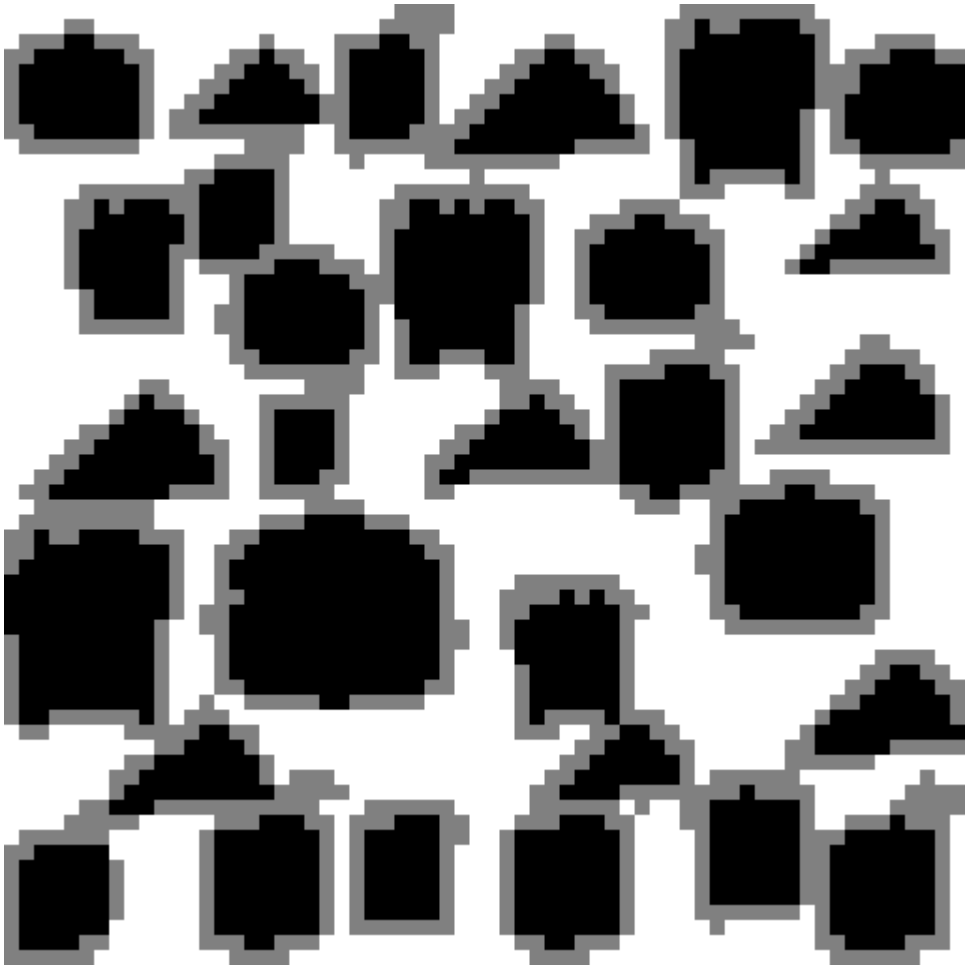
2.png

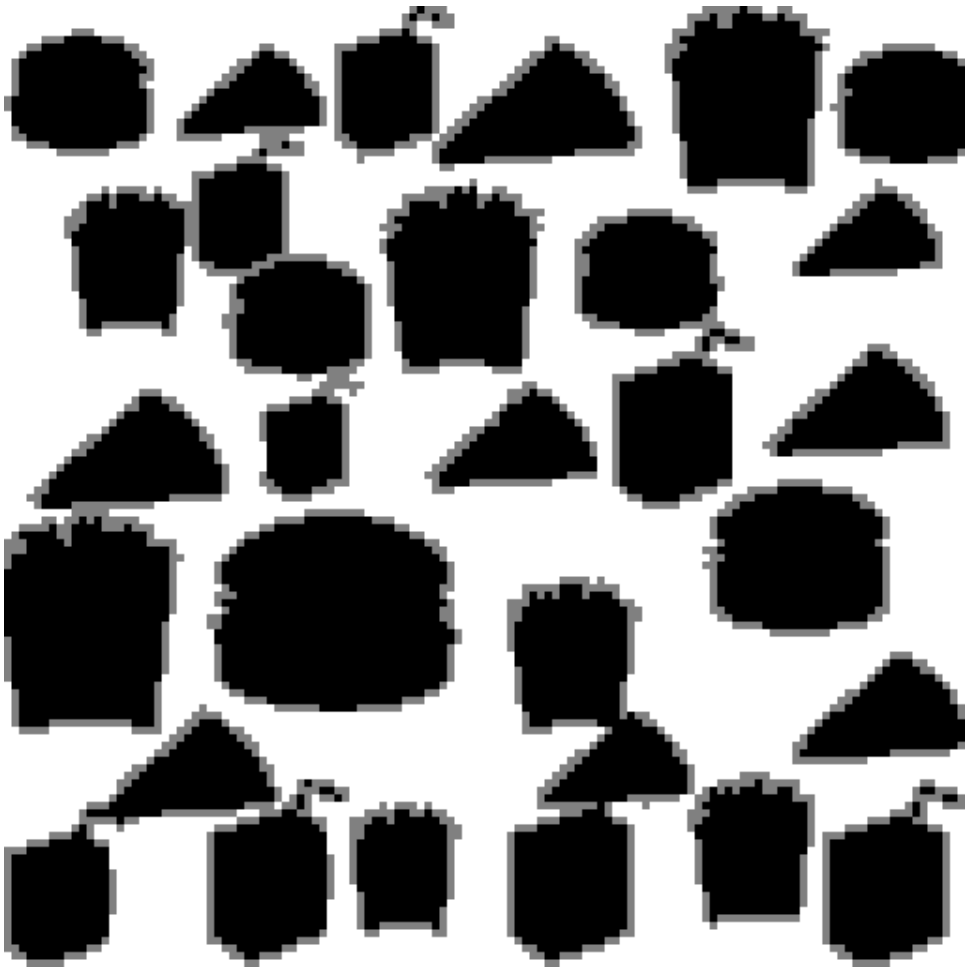


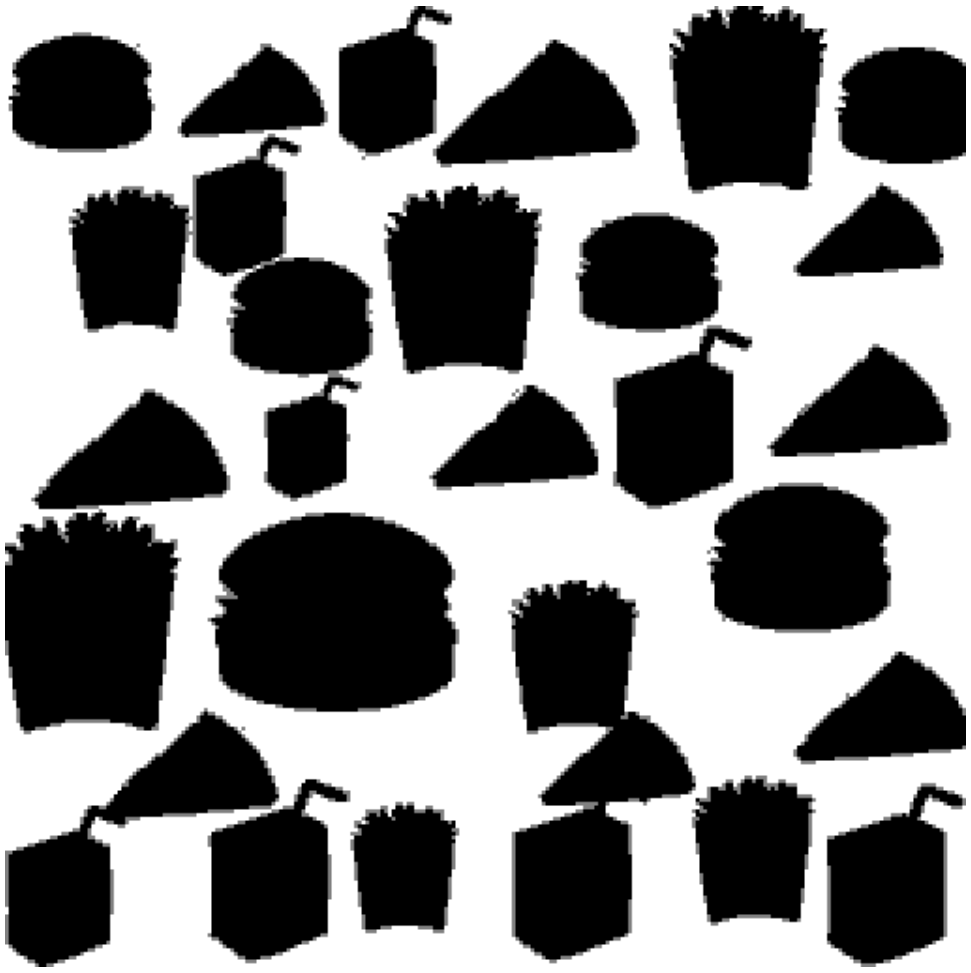


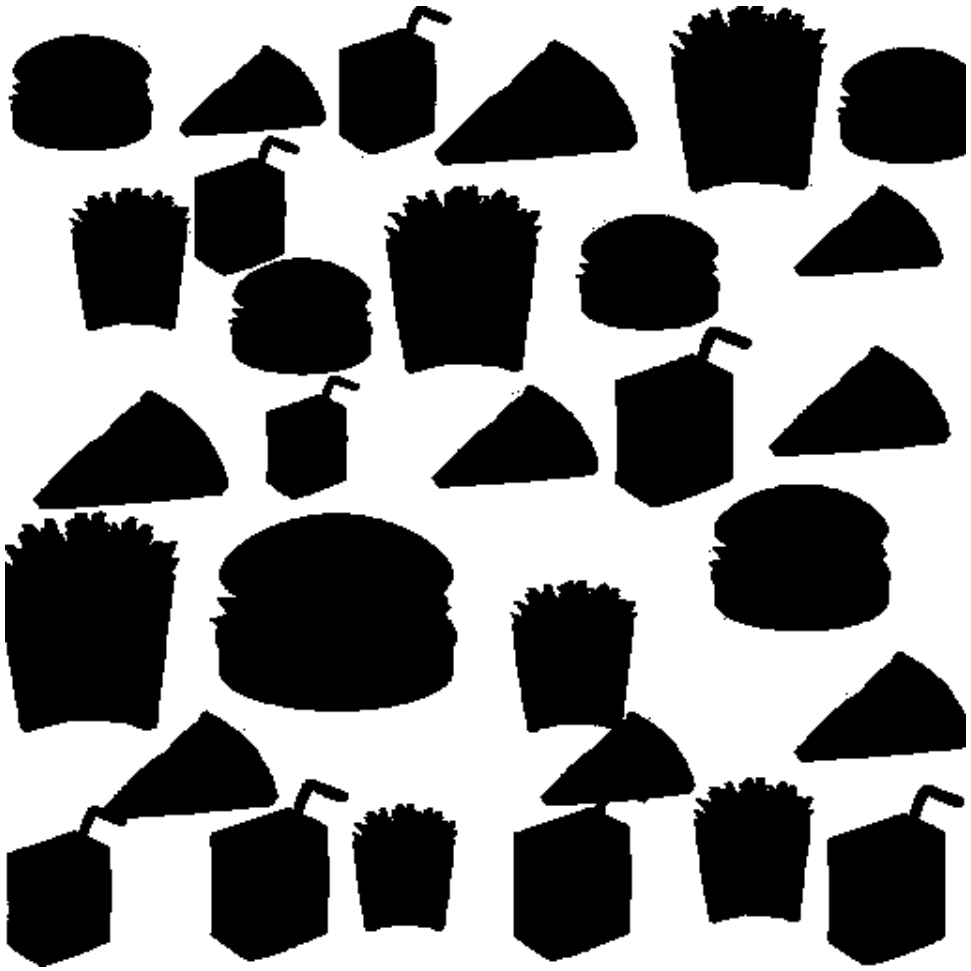




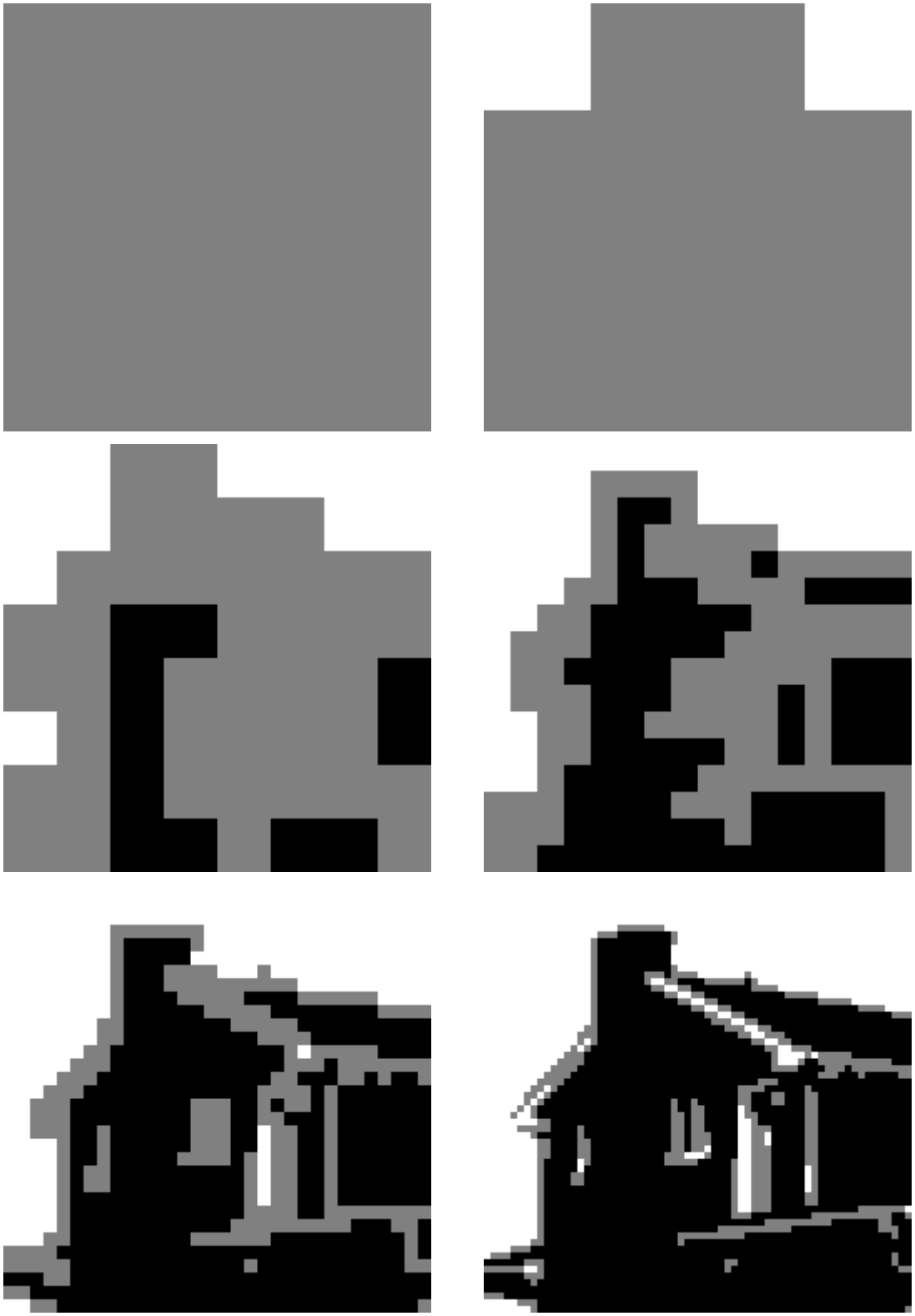








3.png



4.png