機器視覺 HW2 資工三 109590041 范遠皓

環境

Visual Studio 2019、C++ 、openCV2.4.13.6

Component Labeling

- Convert the color image to a binary image

同上次作業，根據 threshold 值來進行二值化。

ImageLibrary.cpp ImageLibrary.h Main.cpp

ImageModel → image_model::ImageLibrary ConvertToG

```cpp
// 二值化
Mat ImageLibrary::ConvertToBinary(Mat colorImage, uchar threshold) {
    Mat grayImage = this->ConvertToGray(colorImage);
    Mat binaryImage(grayImage.size(), CV_8UC1);

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

圖片對應 threshold 值設定

ImageLibrary.cpp ImageLibrary.h Main.cpp

ImageModel

```cpp
int main() {
    // image file names
    const string IMAGE_NAMES[] = {
        "1",
        "2",
        "3",
        "4",
    };

    // threshold 設定
    const int THRESHOLD_SETTRING[] = {
        128,
        210,
        80,
        210,
    };
```

• Labeling components using 4-connected and 8-connected.

ConvertToLabeling () 參數:

binaryImage : 二值化影像

connected : 連通數

objNumber : 將 label 的物件數量寫入指標

sizeFilter : Size Filtering 大小

```
ImageLibrary.cpp   ImageLibrary.h   Main.cpp
ImageModel                          (全域範圍)
class ImageLibrary
{
public:
    ImageLibrary() {};

    // 連通數 4-connected 8-connected
    enum class Connected
    {
        Four = 4,
        Eight = 8
    };

    // 轉灰階
    Mat ConvertToGray(Mat colorImage);
    // 二值化
    Mat ConvertToBinary(Mat colorImage, uchar threshold = 128);
    // Labeling Image，connected: 連通數，objNumber: 寫入 label 的物件數量，sizeFilter: Size Filtering
    Mat ConvertToLabeling(Mat binaryImage, Connected connected = Connected::Four, int* objNumber = nullptr, int sizeFilter = -1);
```

1. 將 uchar 改為 int 並將 0 變成 -1 (物件) ， 255 變成 0 (背景)

```
ImageLibrary.cpp   ImageLibrary.h   Main.cpp
ImageModel                          image_model::ImageLibrary          ConvertToLabeling(Ma
// binaryImage 轉 Labeling Image
Mat ImageLibrary::ConvertToLabeling(Mat binaryImage, Connected connected, int* objNumber, int sizeFilter){
    // 將 uchar 改為 int 並將 0 變成 -1 (物件) ， 255 變成 0 (背景)
    Mat labels(binaryImage.size(), CV_32S);
    for (int row = 0; row < binaryImage.rows; row++)
        for (int col = 0; col < binaryImage.cols; col++)
            labels.at<int>(row, col) = binaryImage.at<uchar>(row, col) > 0 ? 0 : -1;
```

2. 使用 Recursive Algorithm 方式進行 label 同時記錄 object size

ImageLibrary.cpp | ImageLibrary.h | Main.cpp

ImageModel → image_model::ImageLibrary → ConvertToLabeling(Mat binaryImage, Connected conn

```cpp
// labeling (Recursive Algorithm)
vector<Point> checkPoints = { Point(0, -1), Point(-1, 0), Point(0, 1), Point(1, 0) }; // 4-connected 的 checkPoint
vector<Point> eightConnected = { Point(-1, -1), Point(-1, 1), Point(1, -1), Point(1, 1) }; // 8-connected 需加入的 checkPoint
if (connected == Connected::Eight)
    checkPoints.insert(checkPoints.end(), eightConnected.begin(), eightConnected.end());
auto InImage = [rows = labels.rows, cols = labels.cols](Point point) { // 檢查 point 座標是否合法
    return point.x >= 0 && point.y >= 0 && point.x < rows && point.y < cols;
};
vector<int> objSize = { 0 }; // 紀錄每個 object 的大小
int label = 0; // 物件數量
for (int row = 0; row < labels.rows; row++)
    for (int col = 0; col < labels.cols; col++)
        if (labels.at<int>(row, col) == -1) {
            labels.at<int>(row, col) = ++label;
            objSize.push_back(1);
            std::queue<Point> queue;
            queue.push(Point(row, col));
            while (!queue.empty()) {
                Point point = queue.front();
                queue.pop();
                for (Point checkPoint : checkPoints) {
                    checkPoint.x += point.x;
                    checkPoint.y += point.y;
                    if (InImage(checkPoint) && labels.at<int>(checkPoint.x, checkPoint.y) == -1) {
                        queue.push(checkPoint);
                        labels.at<int>(checkPoint.x, checkPoint.y) = label;
                        objSize[label]++;
                    }
                }
            }
        }
```

3. 為每個大小大於 sizeFilter 的物件填色

```cpp
// 為每個物件填色
const int MAX_COLOR = 256 * 256 * 256;
Mat labelingImage(binaryImage.size(), CV_8UC3);
for (int row = 0; row < binaryImage.rows; row++)
    for (int col = 0; col < binaryImage.cols; col++) {
        int objLabel = labels.at<int>(row, col);
        int color = objLabel * (MAX_COLOR / (label + 1));
        labelingImage.at<Vec3b>(row, col) = objLabel > 0 && objSize[objLabel] > sizeFilter ? Vec3b((color >> 16) & 255, (color >> 8) & 255, color & 255) : Vec3b(0, 0, 0
    }
```

4. 將小於 sizeFilter 的物件扣掉

```cpp
// object number
for (int objLabel = 1; objLabel < objSize.size(); objLabel++)
    if (objSize[objLabel] <= sizeFilter)
        label--;
if (objNumber != nullptr)
    *objNumber = label;
```

sizeFilter 設為 100

```cpp
const int SIZE_FILTER = 100;
// 4-connected labeling
int objNumber = 0;
Mat connectedFourImage = library.ConvertToLabeling(binaryImage, ImageLibrary::Connected::Four, &objNumber, SIZE_FILTER);
cv::imshow("4-connected " + IMAGE_PATH, connectedFourImage);
std::cout << "4-connected object number : " << objNumber << '\n';

// 8-connected labeling
Mat connectedEightImage = library.ConvertToLabeling(binaryImage, ImageLibrary::Connected::Eight, &objNumber, SIZE_FILTER);
cv::imshow("8-connected " + IMAGE_PATH, connectedEightImage);
std::cout << "8-connected object number : " << objNumber << '\n';
```

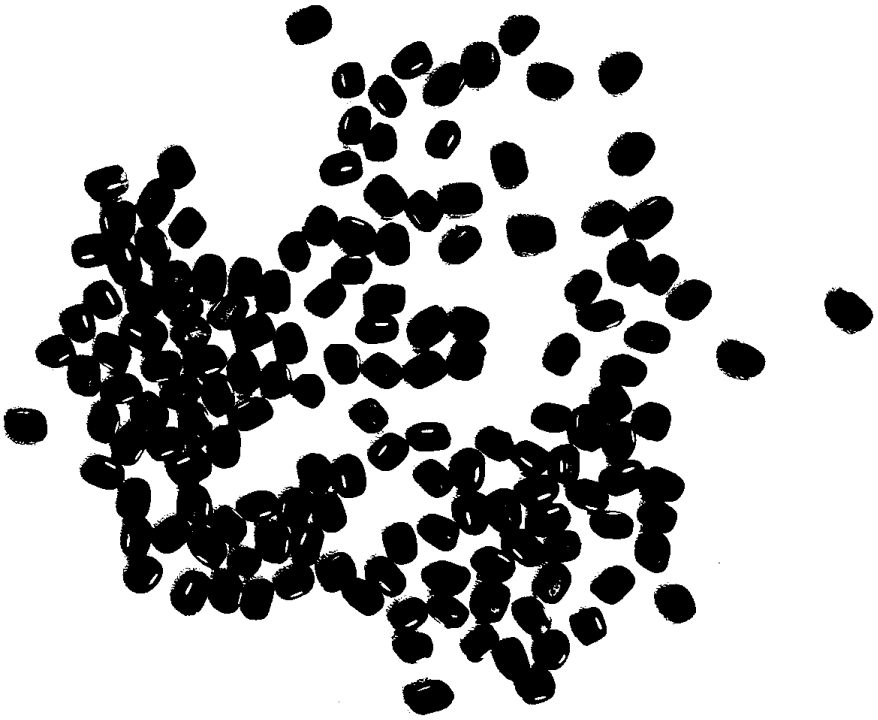
• Output color image and object number.

object number

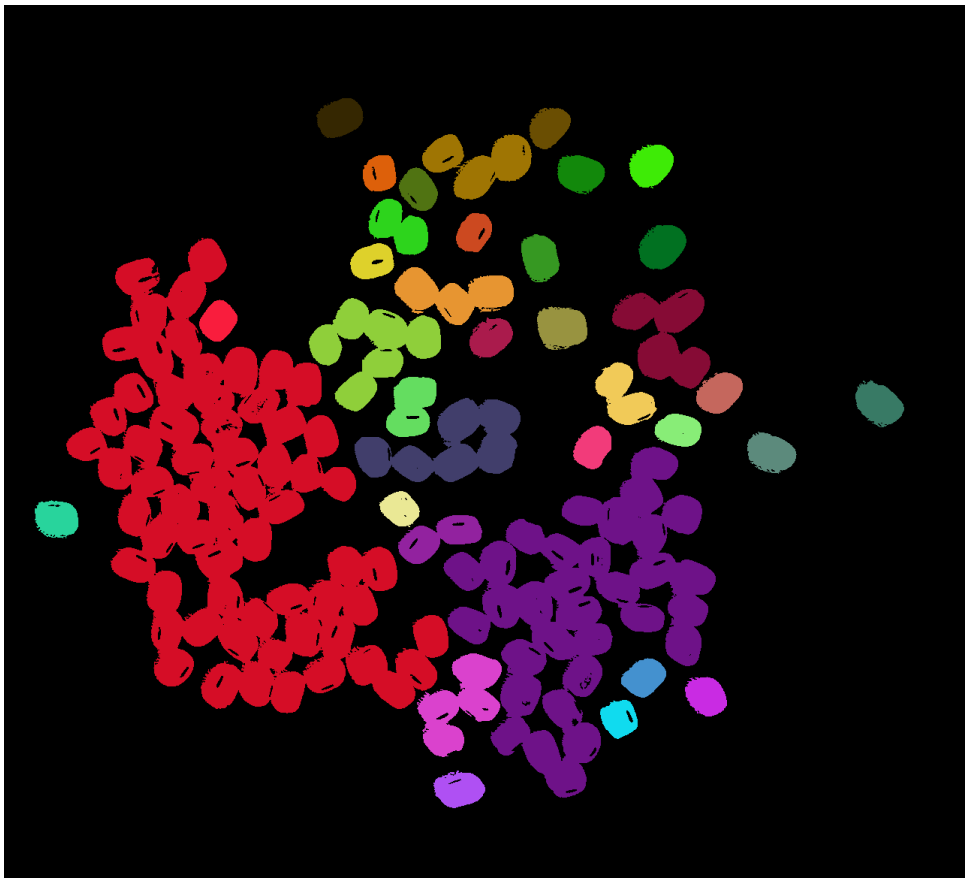
```
Microsoft Visual Studio 偵錯主控台
..\image\1.png
4-connected object number : 36
8-connected object number : 35
..\image\2.png
4-connected object number : 27
8-connected object number : 27
..\image\3.png
4-connected object number : 10
8-connected object number : 10
..\image\4.png
4-connected object number : 26
8-connected object number : 24
```

Output color image

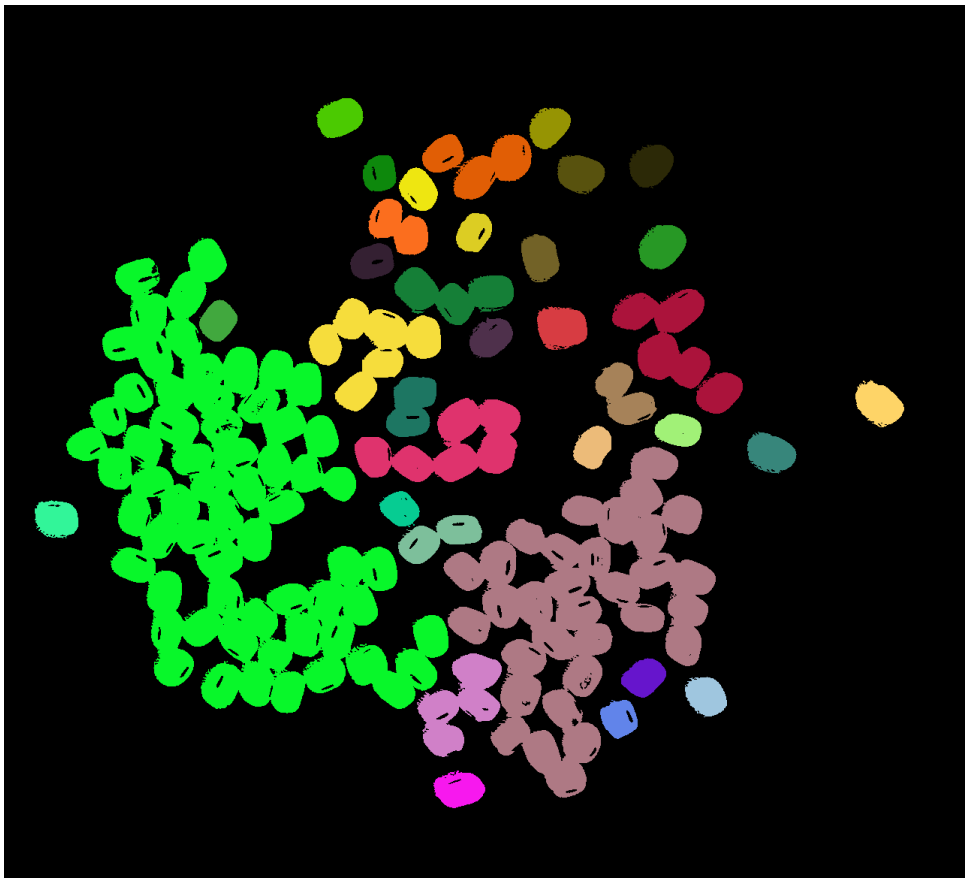
Binary



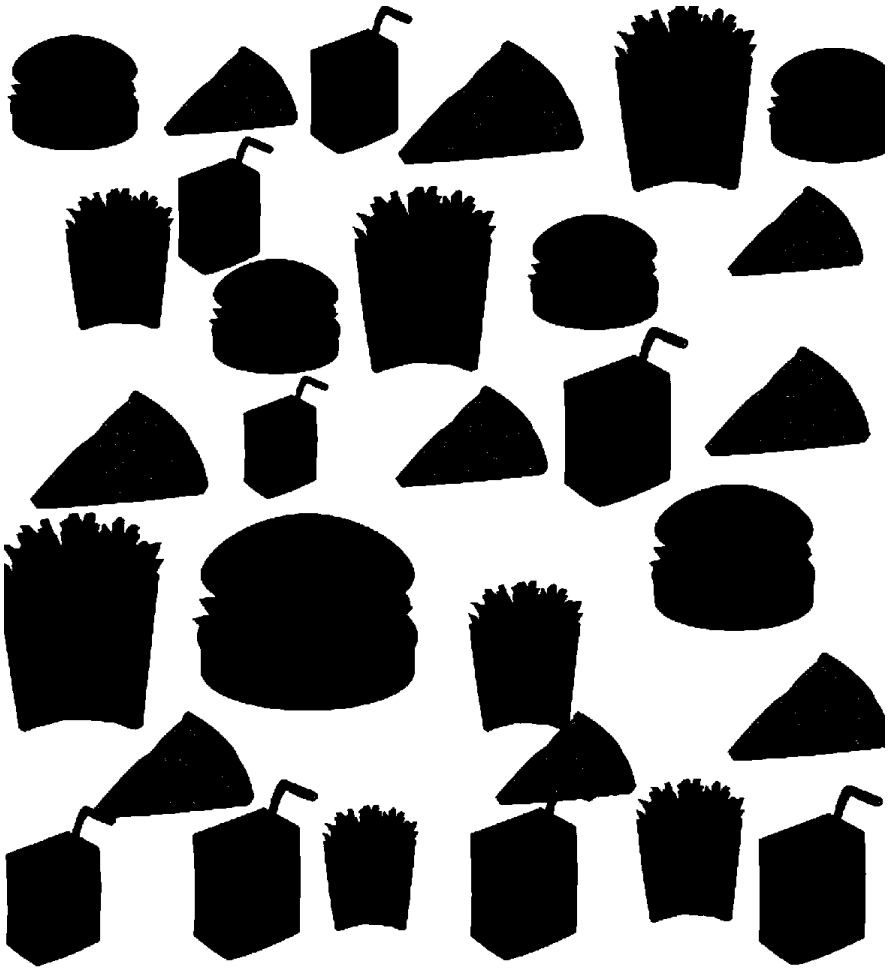
4-connected 36 個



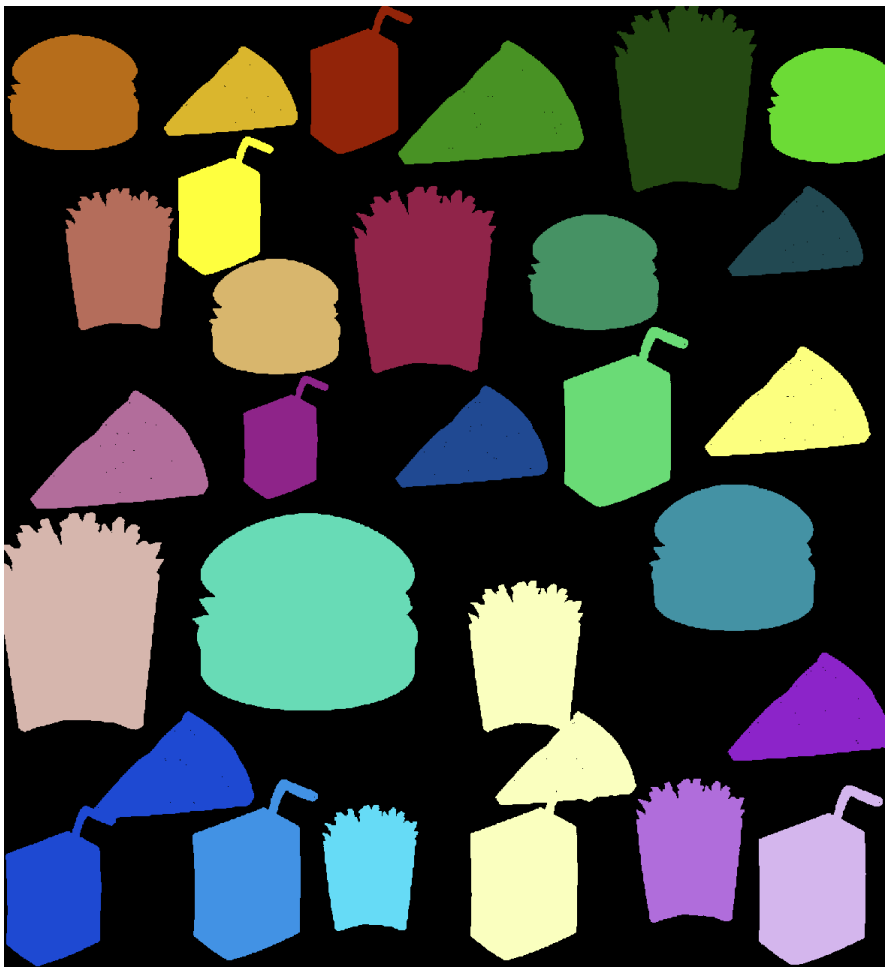
8-connected 35 個



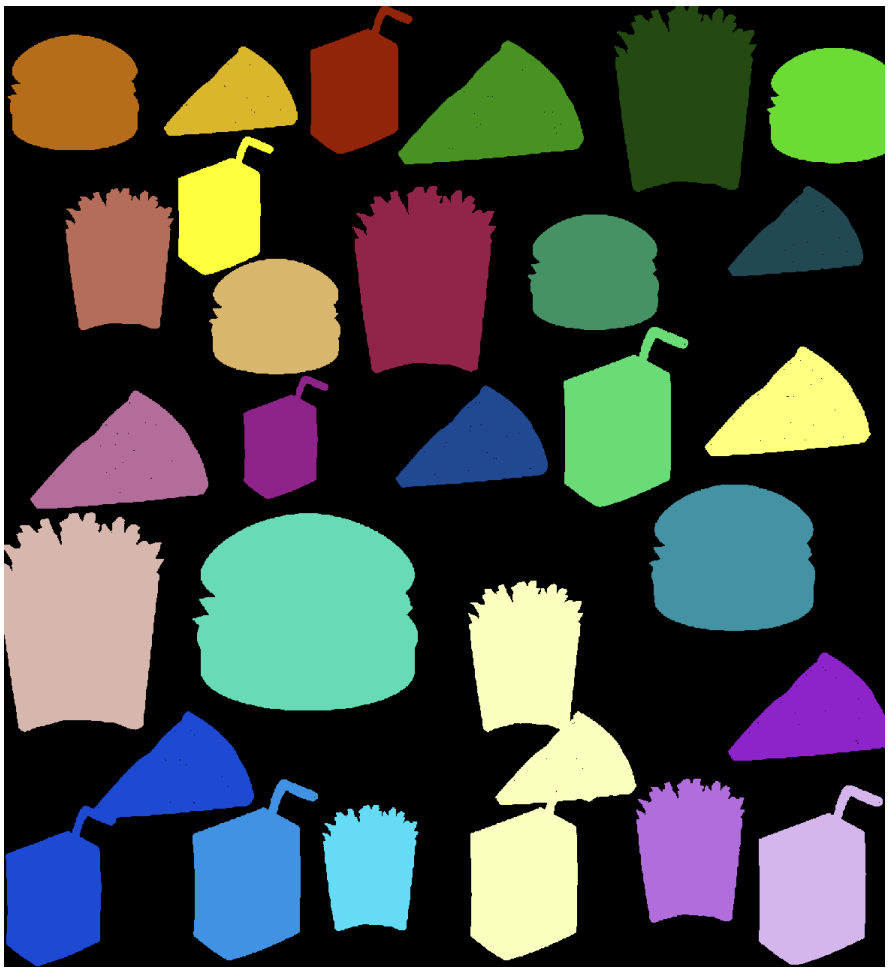
Binary



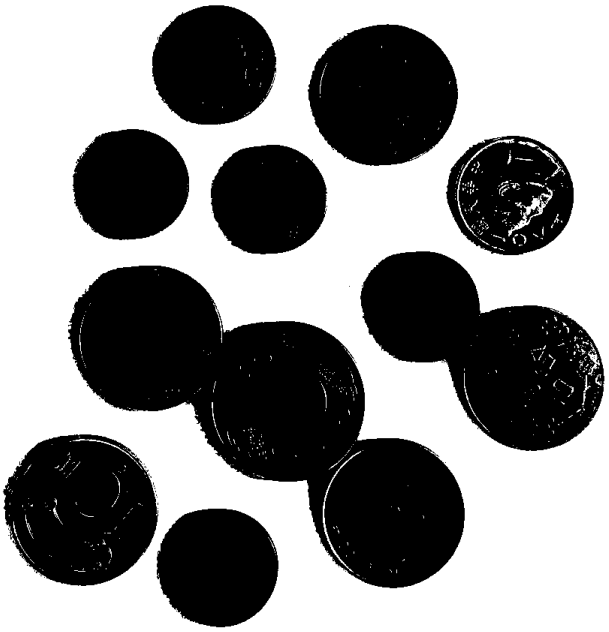
4-connected 27 個



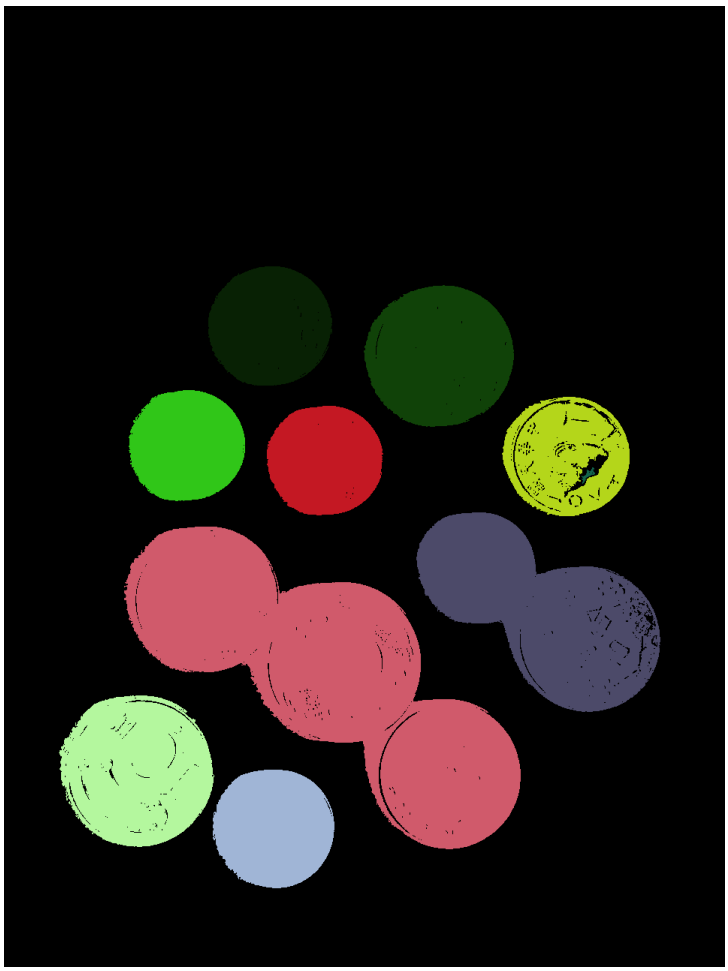
8-connected 27 個



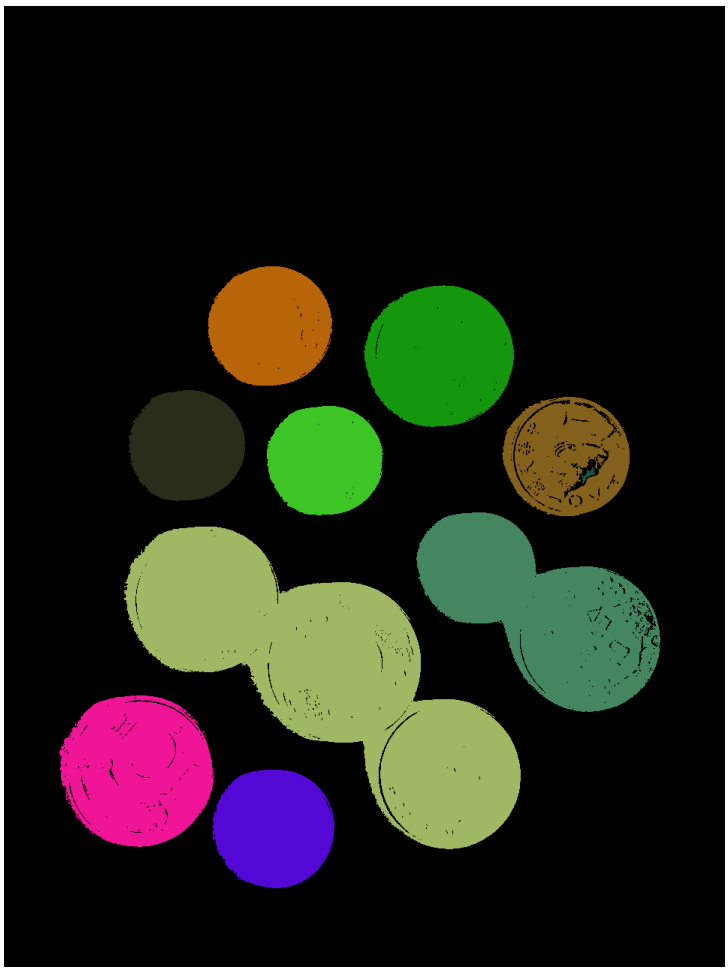
Binary



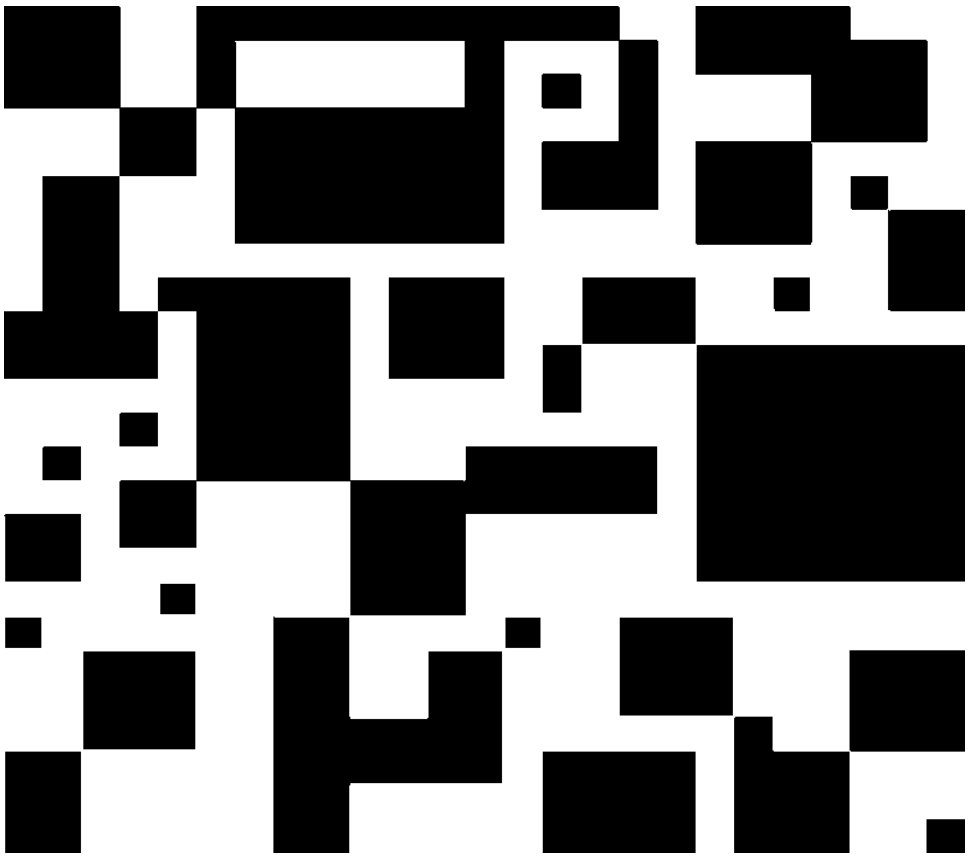
4-connected 10 個



8-connected 10 個



Binary



4-connected 26 個

8-connected 24 個