

|  |
| --- |
| #include <opencv2/opencv.hpp>  #include <iostream>  using namespace cv;  using namespace std;  const String keys =  "{help h usage ? | | print this message }"  "{@input |img0001-spnoise.bmp | 有椒鹽雜訊的灰階影像 }"  "{@lightPattern |lightpattern.bmp | 固定背景影像 }"  ;  Mat Problem1(int argc, char\* argv[])  {  CommandLineParser parser(argc, argv, keys);  Mat image1 = imread(parser.get<String>("@input"));  return image1;  }  Mat Problem2(Mat image1)  {  Mat noiseremoved;  medianBlur(image1, noiseremoved, 5);//可以調值  return noiseremoved;  }  Mat Problem3(int argc, char\* argv[])  {  CommandLineParser parser(argc, argv, keys);  Mat image2 = imread(parser.get<String>("@lightPattern"));  return image2;  }  Mat Problem4(Mat noiseremoved, Mat image2)  {  Mat bgremoved;  absdiff(noiseremoved, image2, bgremoved);  return bgremoved;  }  Mat Problem5(Mat bgremoved)  {  // 對圖像進行二值化處理  Mat thresho;  threshold(bgremoved, thresho, 40, 255, THRESH\_BINARY);//40是自己慢慢試的  //imshow("Threshold", thresho);  return thresho;  }  Mat Problem6(Mat thresho)  {  Mat connectedCompo;  cv::cvtColor(thresho, connectedCompo, cv::COLOR\_RGB2GRAY);//RGB to Gray  Mat bw = (connectedCompo > 40);//將connectedCompo大於40的值，設為255，其餘的值設為0  Mat labelImage(connectedCompo.size(), CV\_32S);  int nLabels = connectedComponents(bw, labelImage, 8);  std::vector<Vec3b> colors(nLabels);  colors[0] = Vec3b(0, 0, 0);//background  for (int label = 1; label < nLabels; ++label) {  colors[label] = Vec3b((rand() & 255), 0, (rand() & 255));  }  Mat dst5(connectedCompo.size(), CV\_8UC3);  for (int r = 0; r < dst5.rows; ++r) {  for (int c = 0; c < dst5.cols; ++c) {  int label = labelImage.at<int>(r, c);  Vec3b& pixel = dst5.at<Vec3b>(r, c);  pixel = colors[label];  }  }  //imshow("Connected Components", dst5);  return dst5;  }  Mat Problem7\_2(Mat img)  {  // 備份原始圖像，用來顯示查找的輪廓  Mat dst = img.clone();  // 彩色圖像轉換為灰度圖像  cvtColor(img, dst, COLOR\_BGR2GRAY);  // 對圖像進行二值化處理  threshold(dst, dst, 40, 255, THRESH\_BINARY\_INV);  std::vector<std::vector<Point> > contours;  std::vector<Vec4i> hierarchy;  // 查找圖像的輪廓 查找的輪廓為白色區域的輪廓  findContours(dst, contours, hierarchy, RETR\_TREE, CHAIN\_APPROX\_SIMPLE);  // 定義外接矩形  std::vector<Rect> boundRect(contours.size());  dst = img.clone();  for (int i = 0; i < contours.size(); i++)  {  // 獲取最小外接矩形  boundRect[i] = boundingRect(contours[i]);  // 在圖像上繪製最小外接矩形  rectangle(dst, boundRect[i], Scalar(0, 255, 0));  }  return dst;  }  int main(int argc, char\* argv[])  {  /\*問題一 : 讀入"img0001-spnoise.bmp"，並顯示在"Input"視窗。\*/  Mat image1 = Problem1(argc, argv);  /\*問題二 : 利用medianBlur()對"img0001-gray.bmp"進行椒鹽雜訊移除，並顯示在"Noise removed"視窗。\*/  Mat noiseremoved = Problem2(image1);  /\*問題三 : 讀入"lightpattern.bmp" ，並顯示在" Background "視窗。\*/  Mat image2 = Problem3(argc, argv);  /\*問題四 : 利用absdiff()與背景影像相減，進行背景移除，並顯示在"BG removed"視窗。\*/  Mat bgremoved = Problem4(noiseremoved, image2);  /\*問題五 : 利用threshold()進行二值化處理，並顯示在"Threshold"視窗。\*/  Mat thresho = Problem5(bgremoved);  /\*問題六 : 利用connectedComponents()進行不同物件範圍分割標定，並顯示在" connectedComponents"視窗。\*/  Mat connectedCo = Problem6(thresho);  /\*問題七 : 利用findContours()找出每個物件的輪廓，再將包含每個物件的輪廓的最小矩形畫在原始圖上，並顯示在"Result"視窗。  \*///老師說不用做  //Mat result = Problem7\_2(connectedCo);  vector <Mat> name = { image1,noiseremoved,image2,bgremoved,thresho,connectedCo};  vector <String> Filename = { "Input","Noise removed","Background","BG removed","Threshold","connectedComponents" };    for (int i = 0; i < 6; i++)  {  ostringstream ss;  ss << Filename[i];  if (i < 3)//前三張顯示在上面  {  namedWindow(ss.str());  moveWindow(ss.str(), 640 \* i, 5);  imshow(ss.str(), name[i]);  }  else  {  namedWindow(ss.str());  moveWindow(ss.str(), 640 \* (i-3), 485);  imshow(ss.str(), name[i]);  }    }  //imshow("Result", result);  waitKey(0);  return 0;  } |