**機器視覺**

HW 1

資工三 109590049 林敬翰

**Part 1**

1-1

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

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;  } |

先創建一個同樣大小但是channel只有1個的mat，再用迴圈把原本圖片中每個pixel中的rgb值用公式計算後再傳入gray中。

1-2

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

Code：

|  |
| --- |
| Mat ConvertToBinary(Mat grayimg) {  int threshold = 128;  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;  } |

先創建一個跟灰階圖片同樣大小的mat，再用迴圈把灰階圖片中每個pixel去跟threshold做判定，比threshold大的話輸出255，反之輸出0。

1-3

Color Map

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

Code：

|  |
| --- |
| Mat ConvertToIndexColorImage(Mat img){  Mat ColorMap(1, 256, CV\_8UC3);  for (int i = 0; i < 256; i++) {  uchar\* pixel = ColorMap.ptr<uchar>(0,i);  pixel[0] = i/3; //b  pixel[1] = i; //g  pixel[2] = 255-i; //r  }  Mat gray = ConvertToGray(img);  Mat indexImg = Mat::zeros(img.size(), CV\_8UC3);;  for (int i = 0; i < gray.rows; i++) {  for (int j = 0; j < gray.cols; j++) {  uchar index = gray.at<uchar>(i, j);  uchar\* c = ColorMap.ptr<uchar>(0, index);  indexImg.at<Vec3b>(i, j)[0] = c[0];  indexImg.at<Vec3b>(i, j)[1] = c[1];  indexImg.at<Vec3b>(i, j)[2] = c[2];  }  }  //make the color map more visible  Mat ViualColorMap(16,16, CV\_8UC3);  int counti = 0;  int countj = 0;  for (int i = 0; i < 256; i++) {  if (i % 16 == 0 && i!=0) {  counti++;  }  ViualColorMap.at<Vec3b>(countj, counti) = ColorMap.at<Vec3b>(0, i);  countj++;  if (countj >= 16) {  countj = 0;  }  }  imwrite("output/Q1-3/ColorMap.png", ViualColorMap);  return indexImg;  } |

先創建一個大小為256的Color Map，然後用迴圈產生顏色。之後將圖片轉成灰階並用迴圈去尋找每個pixel，然後找到灰階圖片在Color Map中對應的index後輸出。由於我的Color Map是做成一維的，所以我後面把他轉成了二維後再輸出。

**Part 2**

2-1

|  |  |
| --- | --- |
| 一張含有 建築, 戶外, 磚, 紅色 的圖片  自動產生的描述 | 一張含有 建築, 戶外, 磚 的圖片  自動產生的描述 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Code：

|  |
| --- |
| Mat ResizeTimesTwo(Mat img) {  Mat newImg(img.cols\*2,img.rows\*2, CV\_8UC3);  for (int i = 0; i < newImg.cols; i+=2) {  for (int j = 0; j < newImg.rows; j += 2) {  newImg.at<Vec3b>(i, j) = img.at<Vec3b>(i / 2, j / 2);  newImg.at<Vec3b>(i+1, j) = img.at<Vec3b>(i / 2, j / 2);  newImg.at<Vec3b>(i, j+1) = img.at<Vec3b>(i / 2, j / 2);  newImg.at<Vec3b>(i+1, j + 1) = img.at<Vec3b>(i / 2, j / 2);  }  }  return newImg;  } |

放大兩倍的話，先宣告一個兩倍大小的Mat，之後用迴圈將原始圖片中的pixel，輸進新的圖片中4個pixel的位置。

|  |
| --- |
| Mat ResizeDevideTwo(Mat img) {  Mat newImg(img.cols / 2, img.rows / 2, CV\_8UC3);  for (int i = 0; i < img.cols; i += 2) {  for (int j = 0; j < img.rows; j += 2) {  newImg.at<Vec3b>(i/2, j/2) = img.at<Vec3b>(i, j);  }  }  return newImg;  } |

縮小兩倍的話，先宣告一個1/2倍大小的Mat，之後用迴圈將原始圖片中每4個pixel中的第一個pixel輸進新的圖片中pixel的位置。

2-2

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Code：

|  |
| --- |
| ResizeTimesTwo2(Mat img) {  Mat newImg(img.cols \* 2, img.rows \* 2, CV\_8UC3);  for (int i = 0; i < newImg.cols; i++) {  for (int j = 0; j < newImg.rows; j++) {  float x = ((float)j) / 2.0;  float y = ((float)i) / 2.0;  int x1 = (int)x;  int y1 = (int)y;  int x2 = x1 + 1;  int y2 = y1 + 1;  if (x2 >= img.cols) {  x2 = img.cols - 1;  }  if (y2 >= img.rows) {  y2 = img.rows - 1;  }  float a = x - x1;  float b = y - y1;  //4 point  Vec3b pixel1 = img.at<Vec3b>(y1, x1);  Vec3b pixel2 = img.at<Vec3b>(y1, x2);  Vec3b pixel3 = img.at<Vec3b>(y2, x1);  Vec3b pixel4 = img.at<Vec3b>(y2, x2);  Vec3b pixel;  pixel[0] = (1 - a) \* (1 - b) \* pixel1[0] + a \* (1 - b) \* pixel2[0] + (1 - a) \* b \* pixel3[0] + a \* b \* pixel4[0];  pixel[1] = (1 - a) \* (1 - b) \* pixel1[1] + a \* (1 - b) \* pixel2[1] + (1 - a) \* b \* pixel3[1] + a \* b \* pixel4[1];  pixel[2] = (1 - a) \* (1 - b) \* pixel1[2] + a \* (1 - b) \* pixel2[2] + (1 - a) \* b \* pixel3[2] + a \* b \* pixel4[2];  newImg.at<Vec3b>(i, j) = pixel;  }  }  return newImg;  } |

放大兩倍的話，使用了Bilinear Interpolation，先取得鄰近的4個點，之後再將每個pixel帶入Bilinear Interpolation的公式後輸出。

|  |
| --- |
| Mat ResizeDevideTwo2(Mat img) {  Mat newImg(img.cols / 2, img.rows / 2, CV\_8UC3);  for (int i = 0; i < img.cols; i += 2) {  for (int j = 0; j < img.rows; j += 2) {  newImg.at<Vec3b>(i / 2, j / 2)[0] = (img.at<Vec3b>(i, j)[0] + img.at<Vec3b>(i + 1, j)[0] + img.at<Vec3b>(i, j + 1)[0] + img.at<Vec3b>(i + 1, j + 1)[0]) / 4;  newImg.at<Vec3b>(i / 2, j / 2)[1] = (img.at<Vec3b>(i, j)[1] + img.at<Vec3b>(i + 1, j)[1] + img.at<Vec3b>(i, j + 1)[1] + img.at<Vec3b>(i + 1, j + 1)[1]) / 4;  newImg.at<Vec3b>(i / 2, j / 2)[2] = (img.at<Vec3b>(i, j)[2] + img.at<Vec3b>(i + 1, j)[2] + img.at<Vec3b>(i, j + 1)[2] + img.at<Vec3b>(i + 1, j + 1)[2]) / 4;  }  }  return newImg;  } |

縮小兩倍的話，先宣告一個1/2倍大小的Mat，之後用迴圈將原始圖片中每4個pixel算出平均值後輸入進新的圖片中pixel的位置。