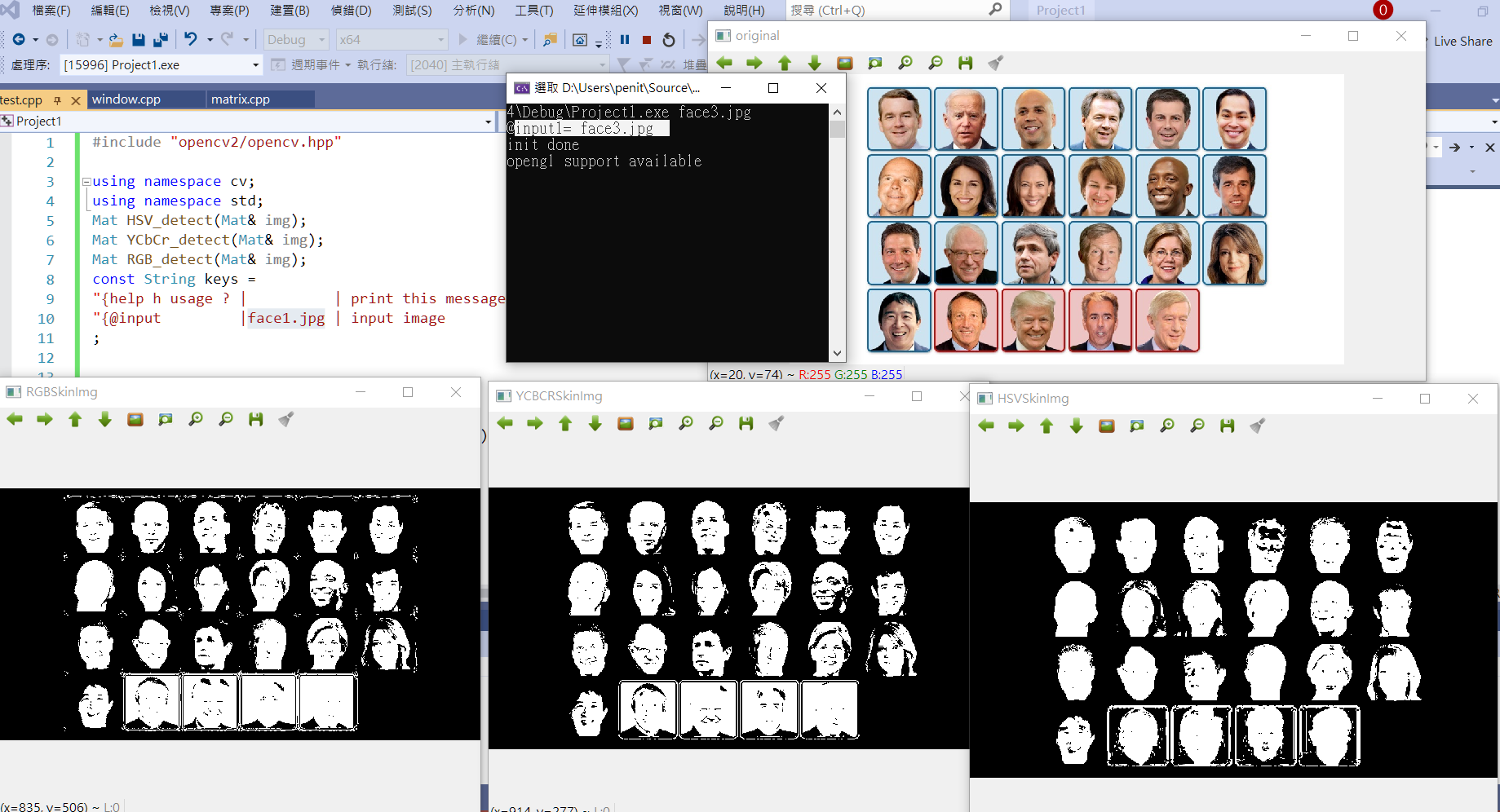
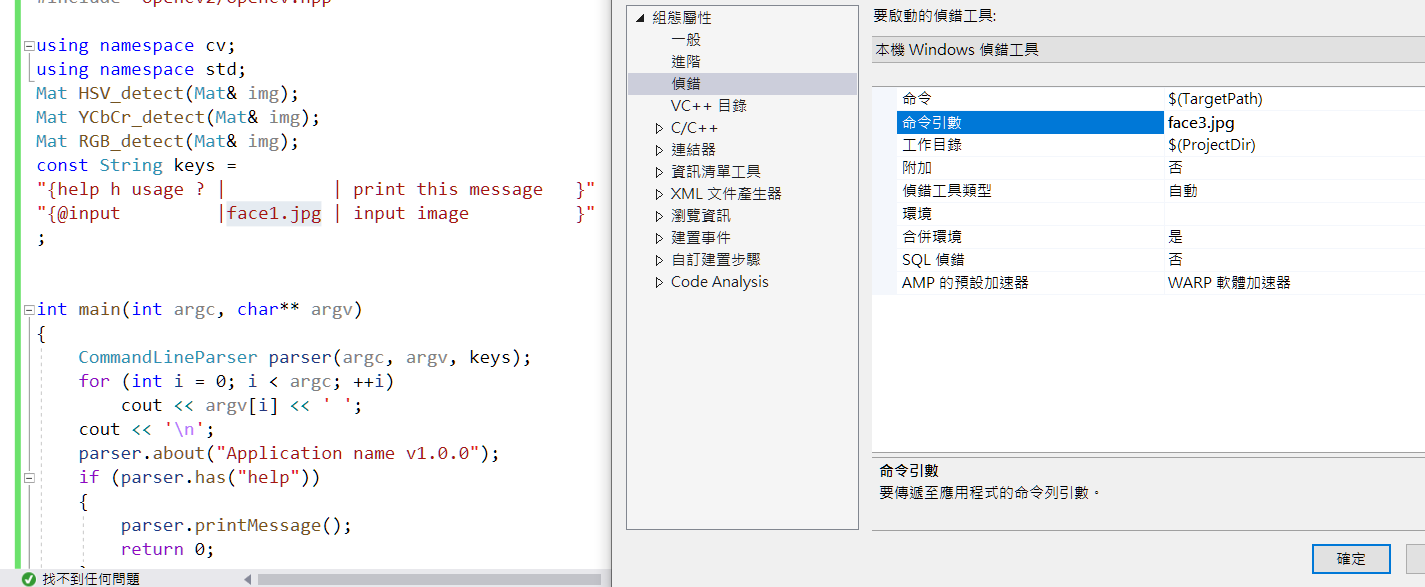
* 將原始彩色影像顯示在一個名為” original ” 視窗
* 將RGB色彩空間的膚色檢測結果顯示在一個名為” rgbSkinImg” 視窗
* 將YCbCr色彩空間的膚色檢測結果顯示在一個名為” ycbcrSkinImg” 視窗
* 將HSV色彩空間的膚色檢測結果顯示在一個名為” hsvSkinImg” 視窗
* 分別針對”face3.jpg”進行測試，並將測試結果剪貼至word檔。
* 將完成的原始碼”lab3-你的學號.cpp”及”lab3-你的學號.docx”檔上傳至moodle





|  |
| --- |
| #include "opencv2/opencv.hpp"  using namespace cv;  using namespace std;  Mat HSV\_detect(Mat& img);  Mat YCbCr\_detect(Mat& img);  Mat RGB\_detect(Mat& img);  const String keys =  "{help h usage ? | | print this message }"  "{@input |face1.jpg | input image }"  ;  int main(int argc, char\*\* argv)  {  CommandLineParser parser(argc, argv, keys);  for (int i = 0; i < argc; ++i)  cout << argv[i] << ' ';  cout << '\n';  parser.about("Application name v1.0.0");  if (parser.has("help"))  {  parser.printMessage();  return 0;  }    cout << "@input1= " << parser.get<String>(0) << endl;    if (!parser.check())  {  parser.printErrors();  return 0;  }  Mat colorImg = imread(parser.get<String>(0));  if (!colorImg.data) // Check for invalid input  {  cout << "Could not open or find the image" << std::endl;  return -1;  }  // Create windows  namedWindow("original", WINDOW\_NORMAL);  imshow("original", colorImg);  Mat rgbSkinImg = RGB\_detect(colorImg);  Mat ycbcrSkinImg = YCbCr\_detect(colorImg);  Mat hsvSkinImg = HSV\_detect(colorImg);  // Create windows  namedWindow("RGBSkinImg", WINDOW\_NORMAL);  imshow("RGBSkinImg", rgbSkinImg);  // Create windows  namedWindow("YCBCRSkinImg", WINDOW\_NORMAL);  imshow("YCBCRSkinImg", ycbcrSkinImg);  // Create windows  namedWindow("HSVSkinImg", WINDOW\_NORMAL);  imshow("HSVSkinImg", hsvSkinImg);  waitKey(0);  }  /\* HSV color space  0 ≤ H ≤ 20 && S ≥ 48 && V ≥ 50  \*/  Mat HSV\_detect(Mat& img)  {  Mat hsvImg;  cvtColor(img, hsvImg, COLOR\_BGR2HSV);  Mat output\_mask = Mat::zeros(img.size(), CV\_8UC1);  int h, s, v;  for (int i = 0; i < img.rows; i++)  {  for (int j = 0; j < img.cols; j++)  {  h = hsvImg.at<Vec3b>(i, j)[0];  s = hsvImg.at<Vec3b>(i, j)[1];  v = hsvImg.at<Vec3b>(i, j)[2];  bool e1 = ((0 <= h) && (h <= 20)) && (s >= 48) && (v >= 50);  if (e1)  output\_mask.at<uchar>(i, j) = 255;  }  }  return output\_mask;  }  /\* YCbCr color space  135 < Cr < 180 && 85 < Cb < 135 && 80 < Y  \*/  Mat YCbCr\_detect(Mat& img)  {  Mat ycbcrImg;  cvtColor(img, ycbcrImg, COLOR\_BGR2YCrCb);  Mat output\_mask = Mat::zeros(img.size(), CV\_8UC1);  int y, cb, cr;  for (int i = 0; i < img.rows; i++)  {  for (int j = 0; j < img.cols; j++)  {  y = ycbcrImg.at<Vec3b>(i, j)[0];  cr = ycbcrImg.at<Vec3b>(i, j)[1];  cb = ycbcrImg.at<Vec3b>(i, j)[2];  bool e1 = (135 < cr) && (cr < 180) && (85 < cb) && (cb < 135) && (80 < y);  if (e1)  output\_mask.at<uchar>(i, j) = 255;  }  }  return output\_mask;  }  /\* RGB color space  R>95 AND G>40 B>20 AND MAX(R,G,B)-MIN(R,G,B)>15 AND ABS(R-G)>15 AND R>G AND R>B  OR  R>220 AND G>210 AND B>170 AND ABS(R-G)<=15 AND R>B AND G>B  \*/  Mat RGB\_detect(Mat& img)  {  Mat output\_mask = Mat::zeros(img.rows, img.cols, CV\_8UC1);  if (img.empty() || img.channels() != 3)  {  return output\_mask;  }  for (int i = 0; i < img.rows; i++)  {  for (int j = 0; j < img.cols; j++)  {  int B = img.at<Vec3b>(i, j)[0];  int G = img.at<Vec3b>(i, j)[1];  int R = img.at<Vec3b>(i, j)[2];  bool e1 = (R > 95) && (G > 40) && (B > 20) && ((max(R, max(G, B)) - min(R, min(G, B))) > 15) && (abs(R - G) > 15) && (R > G) && (R > B);  bool e2 = (R > 220) && (G > 210) && (B > 170) && (abs(R - G) <= 15) && (R > B) && (G > B);  if (e1 || e2)  output\_mask.at<uchar>(i, j) = 255;  }  }  return output\_mask;  } |