Advancde computer vision

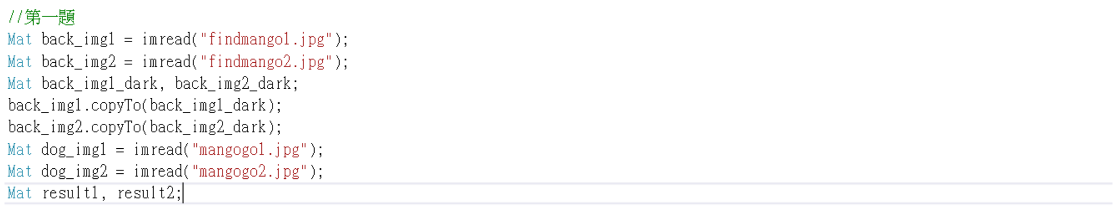
高等電腦視覺

111618018 自動化碩一 吳祐毅

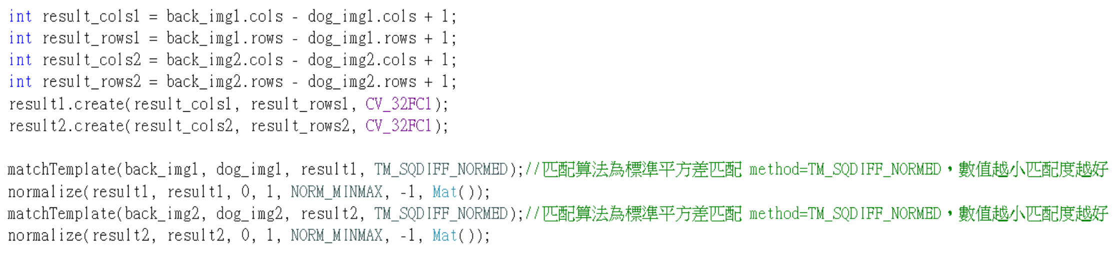
Homework 4

一、 Find Mangogo

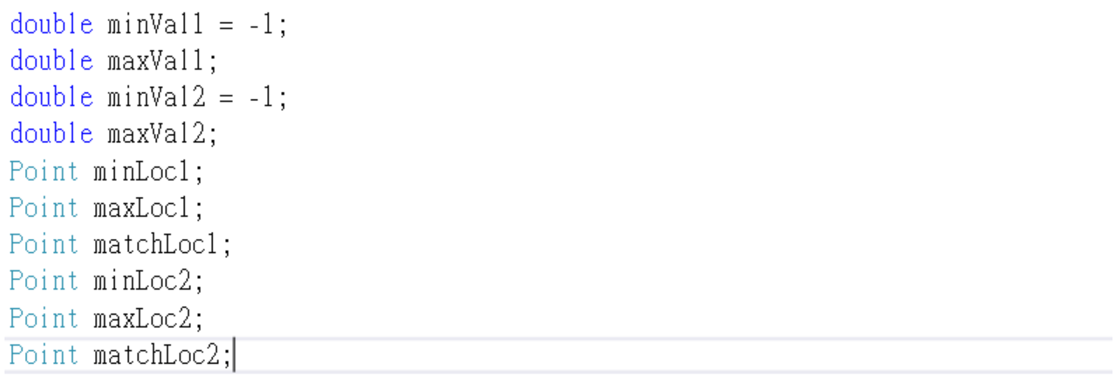
使用 matchTemplate() 和 normalize() 來進行模板匹配，其匹配算法為標準平方差匹配法，最後再透過 minMaxLoc()來尋找匹配的位置。



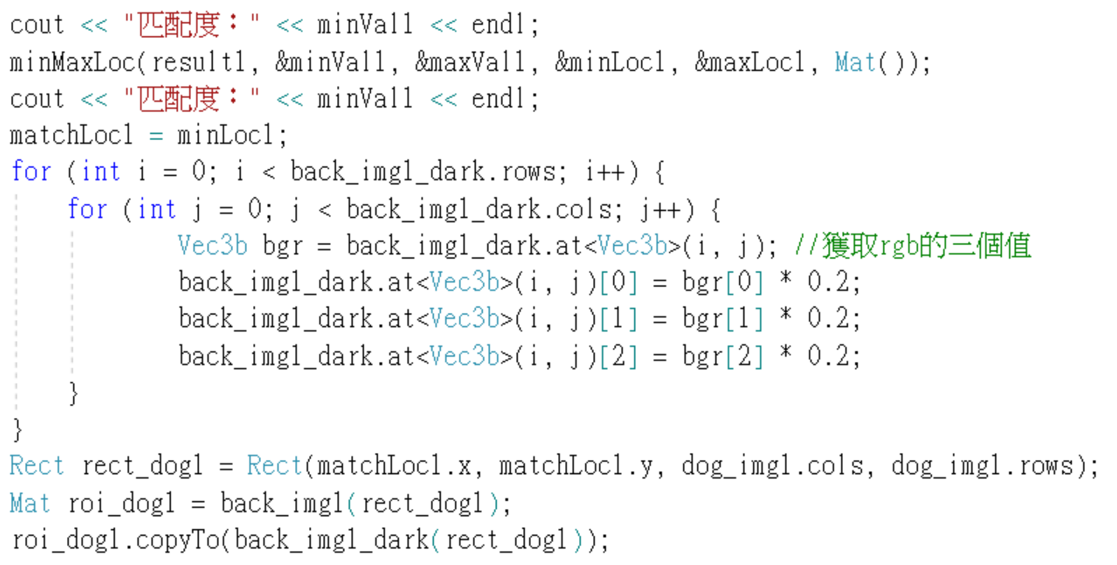
```
//第一題
Mat back_img1 = imread("findmango1.jpg");
Mat back_img2 = imread("findmango2.jpg");
Mat back_img1_dark, back_img2_dark;
back_img1.copyTo(back_img1_dark);
back_img2.copyTo(back_img2_dark);
Mat dog_img1 = imread("mangogo1.jpg");
Mat dog_img2 = imread("mangogo2.jpg");
Mat result1, result2;
```



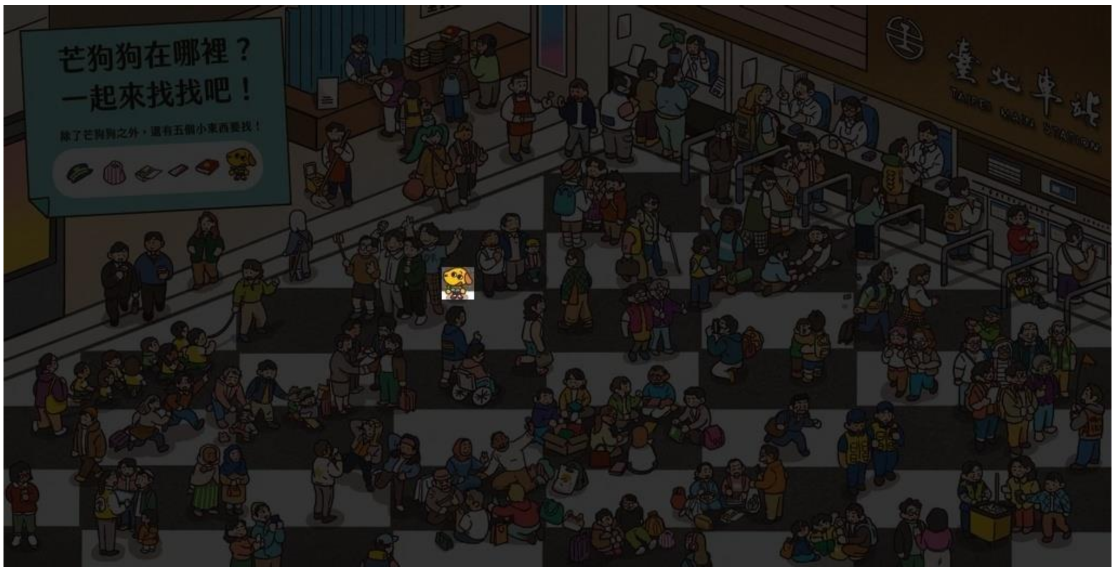
```
int result_cols1 = back_img1.cols - dog_img1.cols + 1;
int result_rows1 = back_img1.rows - dog_img1.rows + 1;
int result_cols2 = back_img2.cols - dog_img2.cols + 1;
int result_rows2 = back_img2.rows - dog_img2.rows + 1;
result1.create(result_cols1, result_rows1, CV_32FC1);
result2.create(result_cols2, result_rows2, CV_32FC1);

matchTemplate(back_img1, dog_img1, result1, TM_SQDIFF_NORMED);//匹配算法為標準平方差匹配 method=TM_SQDIFF_NORMED，數值越小匹配度越好
normalize(result1, result1, 0, 1, NORM_MINMAX, -1, Mat());
matchTemplate(back_img2, dog_img2, result2, TM_SQDIFF_NORMED);//匹配算法為標準平方差匹配 method=TM_SQDIFF_NORMED，數值越小匹配度越好
normalize(result2, result2, 0, 1, NORM_MINMAX, -1, Mat());
```

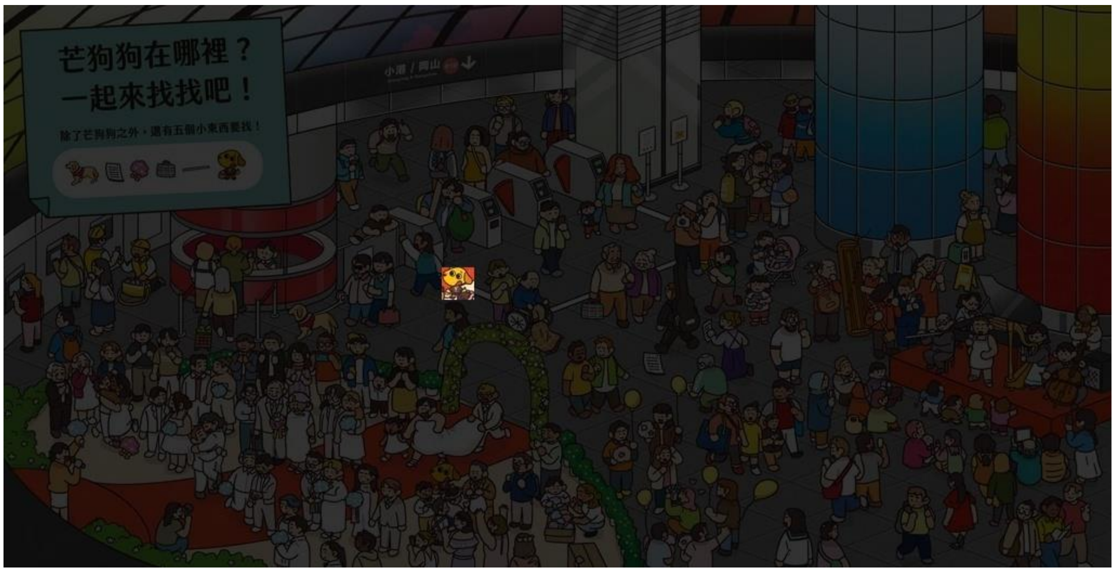


```
double minVal1 = -1;
double maxVal1;
double minVal2 = -1;
double maxVal2;
Point minLoc1;
Point maxLoc1;
Point matchLoc1;
Point minLoc2;
Point maxLoc2;
Point matchLoc2;
```



```
cout << "匹配度：" << minVal1 << endl;
minMaxLoc(result1, &minVal1, &maxVal1, &minLoc1, &maxLoc1, Mat());
cout << "匹配度：" << minVal1 << endl;
matchLoc1 = minLoc1;
for (int i = 0; i < back_img1_dark.rows; i++) {
    for (int j = 0; j < back_img1_dark.cols; j++) {
        Vec3b bgr = back_img1_dark.at<Vec3b>(i, j); //獲取rgb的三個值
        back_img1_dark.at<Vec3b>(i, j)[0] = bgr[0] * 0.2;
        back_img1_dark.at<Vec3b>(i, j)[1] = bgr[1] * 0.2;
        back_img1_dark.at<Vec3b>(i, j)[2] = bgr[2] * 0.2;
    }
}
Rect rect_dog1 = Rect(matchLoc1.x, matchLoc1.y, dog_img1.cols, dog_img1.rows);
Mat roi_dog1 = back_img1(rect_dog1);
roi_dog1.copyTo(back_img1_dark(rect_dog1));
```





二、 Find Wally

匹配方法為標準相關匹配，由於要尋找的人物其模板特徵由 3 個區塊組成，而每一個區塊其匹配結果有許多相同的地方，所以需要先將這三個特徵模塊所有匹配結果的位置找出來，在對每一個位置透過給定的條件去進行刪選，最後找出 Wally 的位置。

為了減少運算量，這裡匹配相關程度選擇 0.98，由於人物是由四個特徵組成，所以使用了 4 個 for 迴圈來對每個位置進行判斷。

```cpp
hat_result.create(Size(resultW_hat, resultH_hat), CV_32FC1);
eye_result.create(Size(resultW_eye, resultH_eye), CV_32FC1);
shirt_result.create(Size(resultH_shirt, resultH_shirt), CV_32FC1);

matchTemplate(wally_img, hat_img, hat_result, TM_CCOEFF_NORMED);
matchTemplate(wally_img, eye_img, eye_result, TM_CCOEFF_NORMED);
matchTemplate(wally_img, shirt_img, shirt_result, TM_CCOEFF_NORMED);

Point matchLoc_hat;
vector<Point> hat_all;
int count_hat = 0;
if (hat_result.channels() == 1)
{
    MatIterator_<float> it_begin, it_end;
    for (it_begin = hat_result.begin<float>(), it_end = hat_result.end<float>(); it_begin != it_end; it_begin++)
    {
        if (*it_begin >= 0.98)
        {
            count_hat++;
            matchLoc_hat = it_begin.pos();
            hat_all.push_back(matchLoc_hat);
            //rectangle(wally_img, matchLoc_hat, Point(matchLoc_hat.x + hat_img.cols, matchLoc_hat.y + hat_img.rows), Scalar(255, 0, 0), 2, 8, 0);
        }
    }
}


cout << count_hat << " " << count_eye << " " << count_shirt << endl;
Point pos_all[4];
float hat_left_eye, left_right_eye, shirt_right_eye;
for (int i = 0; i < hat_all.size(); i++) {
    for (int j = 0; j < left_eye_all.size(); j++) {
        for (int k = 0; k < right_eye_all.size(); k++) {
            for (int m = 0; m < shirt_all.size(); m++) {
                hat_left_eye = sqrt(pow((hat_all[i].x - left_eye_all[j].x), 2) + pow((hat_all[i].y - left_eye_all[j].y), 2));
                left_right_eye = sqrt(pow((left_eye_all[j].x - right_eye_all[k].x), 2) + pow((left_eye_all[j].y - right_eye_all[k].y), 2));
                shirt_right_eye = sqrt(pow((shirt_all[m].x - left_eye_all[j].x), 2) + pow((shirt_all[m].y - left_eye_all[j].y), 2));
                if (hat_left_eye < 20 && left_right_eye < 10 && shirt_right_eye < 30) {
                    //rectangle(wally_img, hat_all[i], Point(right_eye_all[k].x + eye_img.cols, shirt_all[m].y + shirt_img.rows), Scalar(128, 128, 128), 2, 8, 0);
                    Rect rect_wally = Rect(hat_all[i].x, hat_all[i].y, right_eye_all[k].x + eye_img.cols - hat_all[i].x, shirt_all[m].y + shirt_img.rows - hat_all[i].y);
                    Mat roi_wally = wally_img(rect_wally);
                    roi_wally.copyTo(wally_img_dark(rect_wally));
                }
            }
        }
    }
}
```