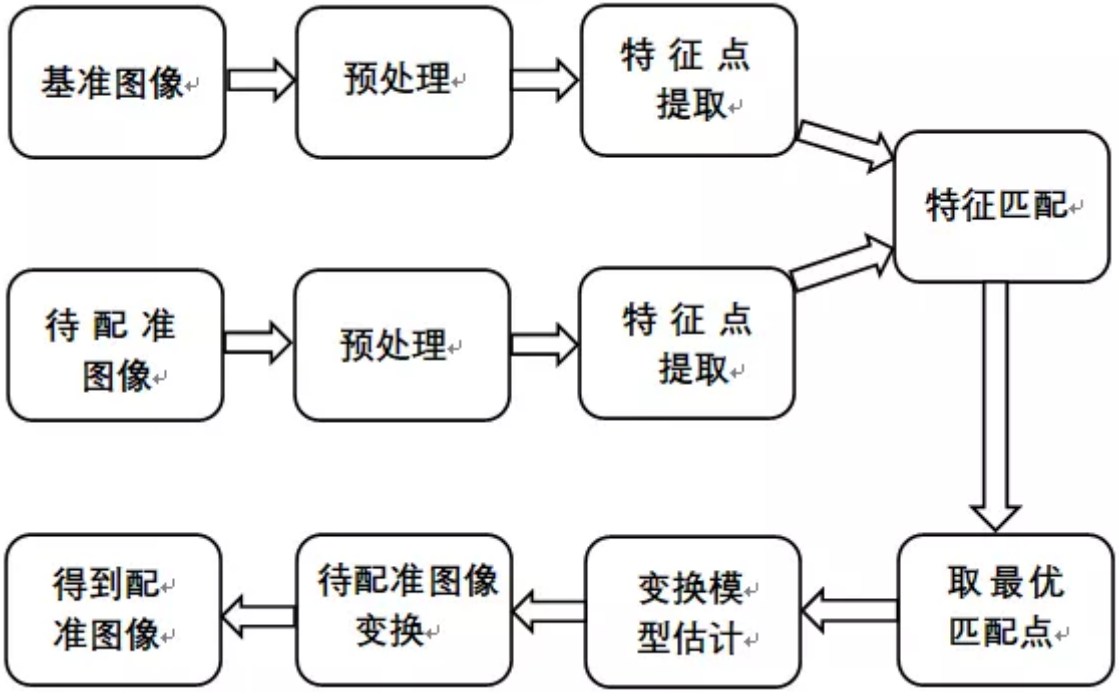
乾貨| 基於特徵的圖像配準用於缺陷檢測

小黃弟 小白學視覺 昨天

特徵提取

基於特徵的圖像配準，具有非常廣泛的應用，大致流程可以如下:



經典的特徵匹配算法有SIFT、SURF、ORB等，這三種方法在OpenCV裡面都已實現。SURF基本就是SIFT的全面升級版，有SURF基本就不用考慮SIFT，而ORB的強點在於計算時間，以下具體比較:

> 計算速度：ORB>>SURF>>SIFT（各差一個量級）

所以結論就是，如果對計算實時性要求非常高，可選用ORB算法，但基本要保證正對拍攝；如果對穩定性要求稍高，可以選擇SURF；基本不用SIFT。此外補充一點，自從OpenCV3.x開始，受到SIFT跟SURF專利授權的影響，OpenCV正式的發布版本中已經移除了SIFT跟SURF算法。ORB特徵提取算法是基於FAST跟BRIEF算法改進的組合算法，其中FAST實現關鍵點/特徵點的檢測，在此基礎上基於幾何矩添加方向屬性，BRIEF實現描述子生成，添加旋轉不變性支持。

ORB特徵匹配速度快的一個原因之一就是使用字符串向量的描述子，避免了浮點數計算。字符串描述子匹配上可以採用漢明距離或者LSH改進算法實現，相比浮點數計算L2距離進一步降低了計算量。所以在一般情況下建議使用ORB特徵匹配，如果效果不好再嘗試AKAZE/SURF/SIFT等其它特徵匹配算法。

特徵對齊/配準

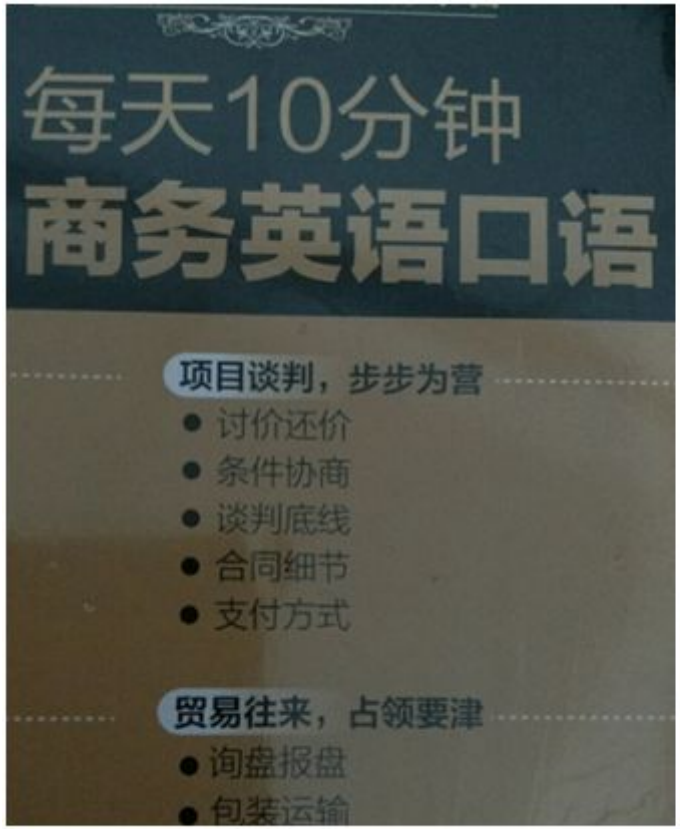
兩幅圖像之間的基於特徵匹配的透視變換矩陣求解通常被稱為圖像對齊或者配準。基於特徵的匹配可以很好實現圖像對齊或者配準，首先需要獲取兩張圖像的特徵關鍵點與特徵描述子，然後通過暴力匹配或者FLANN匹配尋找匹配度高的相關特徵點。最後基於這些相關特徵點估算它們之間的單應性矩陣，通過單應性矩陣實現透視變換，完成圖像對齊與配準。OpenCV中有兩個函數可以獲得單映射變換矩陣，分別為：

```
- findHomography
- getPerspectiveTransform
```

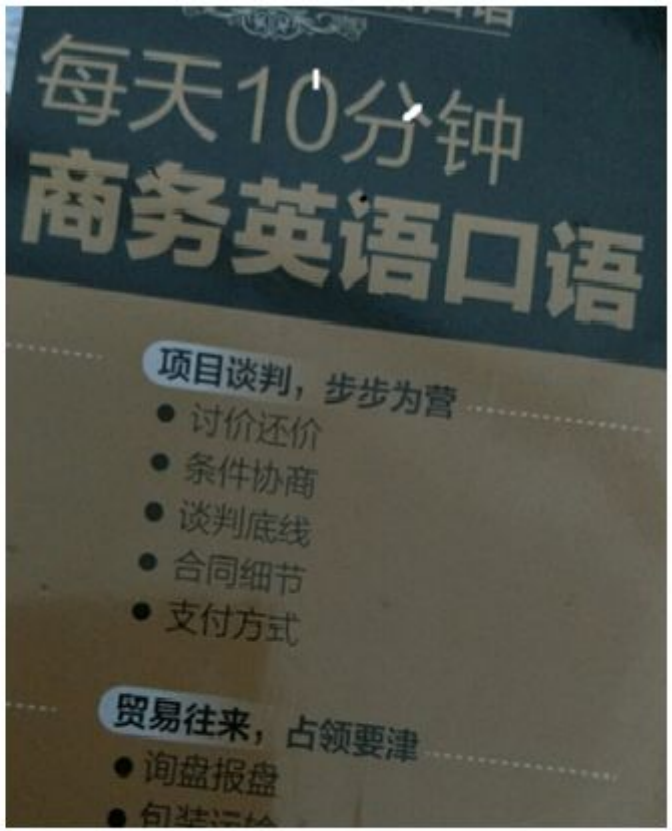
兩者之間的區別在於getPerspectiveTransform只會拿4個點去計算，findHomography則會拿一堆點(>=4)去計算。

應用代碼演示

下面是一個簡單的代碼演示，基於特徵對齊，實現基於分差的缺陷檢測。



基准图像



输入图像

用基於ORB特徵的匹配結果，如下圖所示，可以看到有一些錯誤的匹配點

基於ORB特徵實現圖像相關特徵點匹配的代碼實現如下：

```
constint MAX_FEATURES = 5000;
constfloat GOOD_MATCH_PERCENT = 0.45f;
//im1为待配准图片
//im2为模板图片
//im1Reg为配准后的图片
//h为单应性矩阵
void alignImages(Mat&im1, Mat&im2, Mat&im1Reg, Mat&h)
{
    // 将图像转为灰度图
    Mat im1Gray, im2Gray;
    cvtColor(im1, im1Gray, COLOR_BGR2GRAY);
    cvtColor(im2, im2Gray, COLOR_BGR2GRAY);

    // 存储特征与特征描述子的变量
    std::vector<KeyPoint> keypoints1, keypoints2;
    Mat descriptors1, descriptors2;

    // 检测ORB特征计算特征描述子.
```

```
    Ptr<Feature2D> orb = ORB::create(MAX_FEATURES);
    orb->detectAndCompute(im1Gray, Mat(), keypoints1, descriptors1);
    clock_t start, end;
    start = clock();
    orb->detectAndCompute(im2Gray, Mat(), keypoints2, descriptors2);  //77ms

    // 特征匹配.
    std::vector<DMatch> matches;
    Ptr<DescriptorMatcher> matcher = DescriptorMatcher::create("BruteForce-Hamming");
    matcher->match(descriptors1, descriptors2, matches, Mat());
    // Sort matches by score
    std::sort(matches.begin(), matches.end());

    //基于GMS的特征匹配算法
    //vector<DMatch> matchesAll, matchesGMS;
    //BFMatcher matcher(NORM_HAMMING);
    //std::vector<DMatch> matches;
    //matcher.match(descriptors1, descriptors2, matchesAll);
    //cout << "matchesAll: " << matchesAll.size() << endl;
    //matchGMS(im1.size(), im2.size(), keypoints1, keypoints2, matchesAll, matches);
    //std::sort(matches.begin(), matches.end());

    end = clock();
    cout << (float)(end - start) * 1000 / CLOCKS_PER_SEC<<"ms"<< endl;

    // 移除不好的匹配点
    constint numGoodMatches = matches.size() * GOOD_MATCH_PERCENT;
    matches.erase(matches.begin() + numGoodMatches, matches.end());
    // 画匹配点
    Mat imMatches;
    drawMatches(im1, keypoints1, im2, keypoints2, matches, imMatches);
    imwrite("matches.jpg", imMatches);

    // 存储好的匹配点
    std::vector<Point2f> points1, points2;

    for (size_t i = 0; i < matches.size(); i++)
    {
        points1.push_back(keypoints1[matches[i].queryIdx].pt);
```

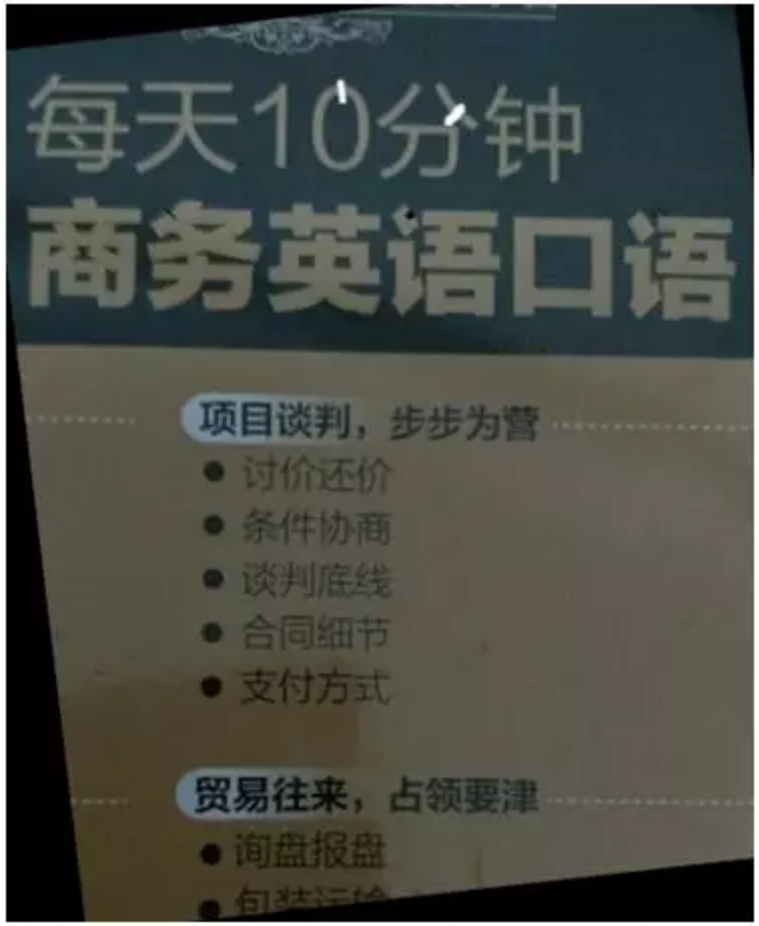
```
        points2.push_back(keypoints2[matches[i].trainIdx].pt);
    }

    // 找出最优单映射变换矩阵h
    h= findHomography(points1, points2, RANSAC);

    // 利用h矩阵进行透视变换
    warpPerspective(im1, im1Reg, h, im2.size());
}
```

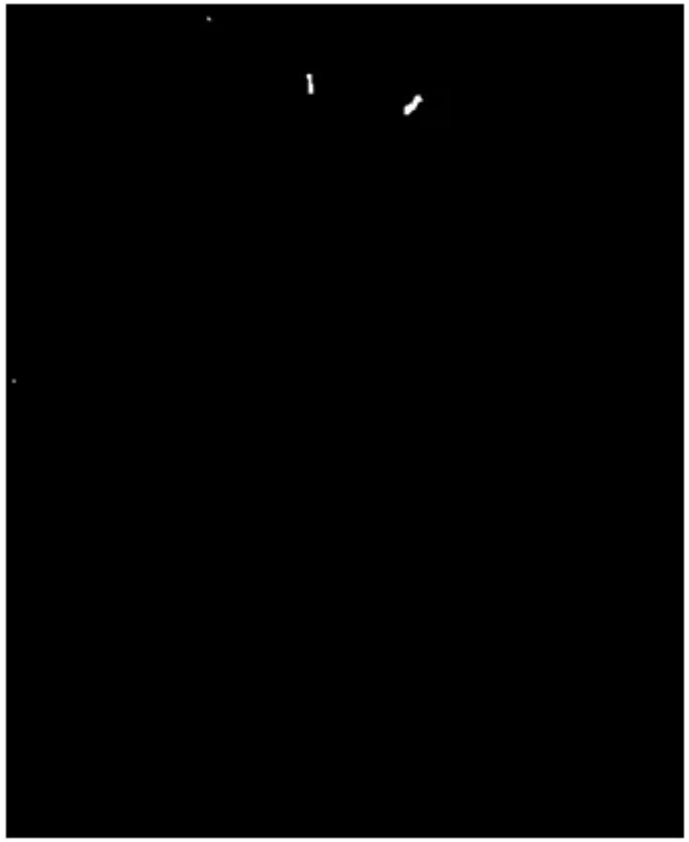
Grid-based Motion Statistics(GMS)通過網格劃分、運動統計特性的方法可以迅速剔除錯誤匹配，以此來提高匹配的穩定性。ORB+GMS的匹配效果如下，可見錯誤的匹配點少了很多。

配准後的圖如下圖所示：

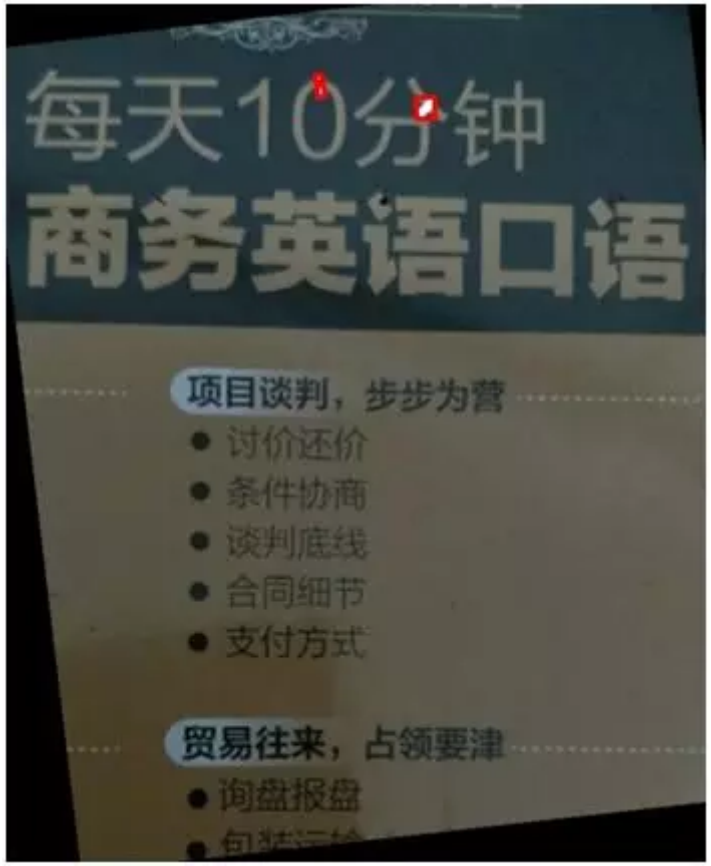


将配准后的图与基准模板图做差分，效果如下：

进行形态学操作，



找出缺陷，比较大的缺陷可以找出来，较小的缺陷还是不能找出来。



这部分的代码实现如下：

```
int main(intargc, char **argv)
{
    // Read reference image
    string refFilename("8.jpg");
    cout <<"Reading reference image : "<< refFilename << endl;
    Mat imReference = imread(refFilename);

    // Read image to be aligned
    string imFilename("7.jpg");
    cout <<"Reading image to align : "<< imFilename << endl;
    Mat im = imread(imFilename);
```

```
// Registered image will be resotred in imReg.
// The estimated homography will be stored in h.
Mat imReg, h;

// Align images
cout <<"Aligning images ..."<< endl;
alignImages(im, imReference, imReg, h);

// Write aligned image to disk.
string outFilename("aligned.jpg");
cout <<"Saving aligned image : "<< outFilename << endl;
imwrite(outFilename, imReg);

// Print estimated homography
cout <<"Estimated homography : \n"<< h << endl;
Mat currentframe, previousframe;
cvtColor(imReference, previousframe, COLOR_BGR2GRAY);
cvtColor(imReg, currentframe, COLOR_BGR2GRAY);  //转化为单通道灰度图

absdiff(currentframe, previousframe, currentframe);//做差求绝对值
imshow("1", currentframe);
imwrite("re.jpg", currentframe);
threshold(currentframe, currentframe, 120, 255.0, THRESH_BINARY);
imwrite("re11.jpg", currentframe);

erode(currentframe, currentframe, Mat());//腐蚀
dilate(currentframe, currentframe, Mat());//膨胀
dilate(currentframe, currentframe, Mat());//膨胀

imshow("moving area", currentframe);    //显示图像

vector<vector<Point>> v;
vector<Vec4i> hierarchy;
Mat result;
Rect rect;
findContours(currentframe, v, hierarchy, RETR_EXTERNAL, CHAIN_APPROX_NONE);
for (int i = 0; i < hierarchy.size(); i++)
{
    rect = boundingRect(v.at(i));
```

```
        if (rect.area() > 1)
        {
            rectangle(imReg, rect, Scalar(0, 0, 255), 2);
        }
    }

    imwrite("res1.jpg", imReg);
    imshow("moving area1", imReg);
    waitKey(0);
}
```

下载1：OpenCV-Contrib扩展模块中文版教程

在「**小白学视觉**」公众号后台回复：**扩展模块中文教程，**即可下载全网第一份OpenCV扩展模块教程中文版，涵盖**扩展模块安装、SFM算法、立体视觉、目标跟踪、生物视觉、超分辨率处理**等二十多章内容。

下载2：Python视觉实战项目52讲

在「**小白学视觉**」公众号后台回复：**Python视觉实战项目，**即可下载包括**图像分割、口罩检测、车道线检测、车辆计数、添加眼线、车牌识别、字符识别、情绪检测、文本内容提取、面部识别**等31个视觉实战项目，助力快速学校计算机视觉。

下载3：OpenCV实战项目20讲

在「**小白学视觉**」公众号后台回复：**OpenCV实战项目20讲，**即可下载含有**20**个基于**OpenCV**实现20个**实战项目**，实现OpenCV学习进阶。

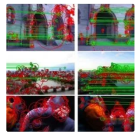
交流群

歡迎加入公眾號讀者群一起和同行交流，目前有 請按照格式備註，否則不予通過 添加成功後會根據研究方向邀請進入相關微信群。請勿

喜歡此內容的人還喜歡

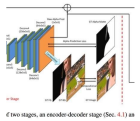
基於深度學習的特徵提取和匹配

小白學視覺



淺談深度學習圖像分割

小白學視覺



解讀基於多傳感器融合的卡爾曼濾波算法

小白學視覺