关于RANSAC的理解 - CSDN博客

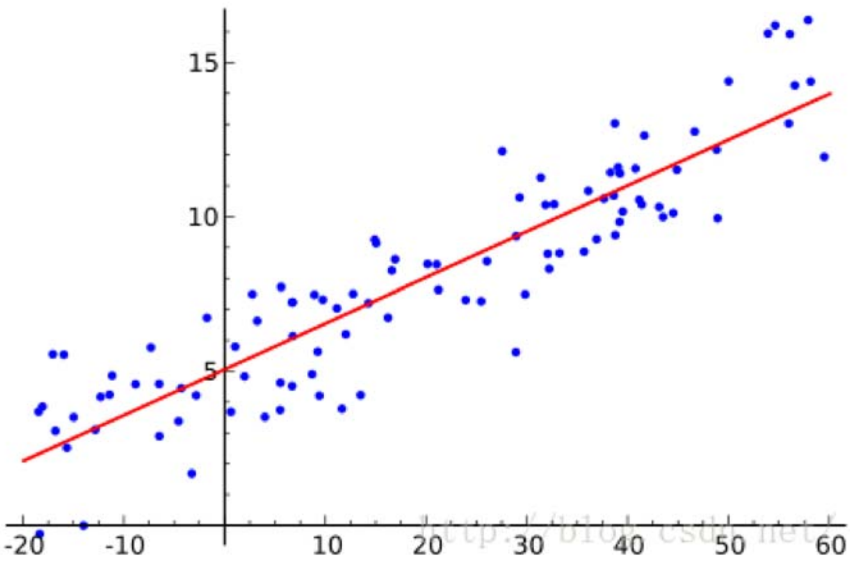
笔记本：	文献
创建时间：	2018/6/4 17:15
URL：	https://blog.csdn.net/zinnc/article/details/52319716

原 关于RANSAC的理解

2016年08月26日 00:35:15 阅读数：5662

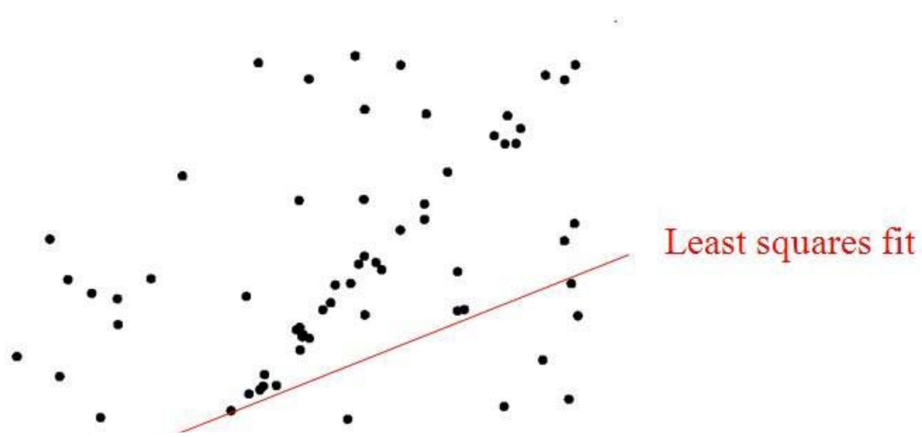
先说最小二乘。

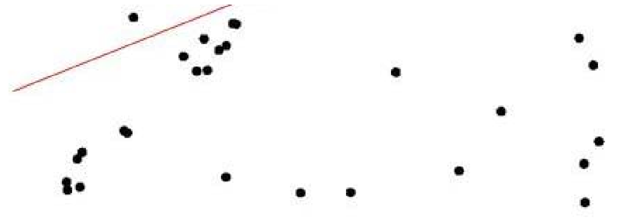
ok，你手头有一堆数据，比如这些蓝点：



那么我们假设它符合一个直线模型：y=ax+b，用最小二乘就可以很容易求解出未知参数a和b。最小二乘大法确实好哇，毕竟高斯用它来估计谷神星的轨道（https://math.berkeley.edu/~mgu/MA221/Ceres_Presentation.pdf；http://www.cnblogs.com/washa/p/3164212.html）。

但是，当你的数据充满了噪声时，比如下面图中的黑点，很明显中间有一条妥妥的直线，但是你也用最小二乘去解它，于是悲剧了：





很显然最小二乘失效了，这时候我们就要用RANSAC去解决它。

RANSAC的使用条件是：

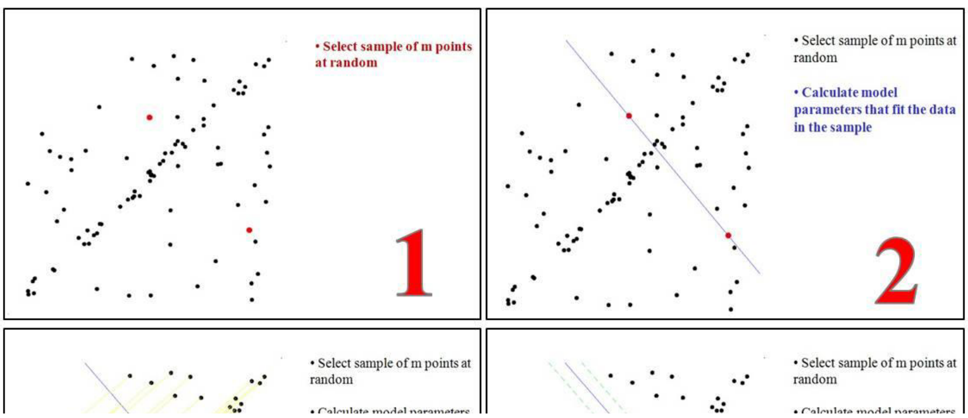
1.输入是一组带污染的观测数据，其中的可信数据占了大多数；

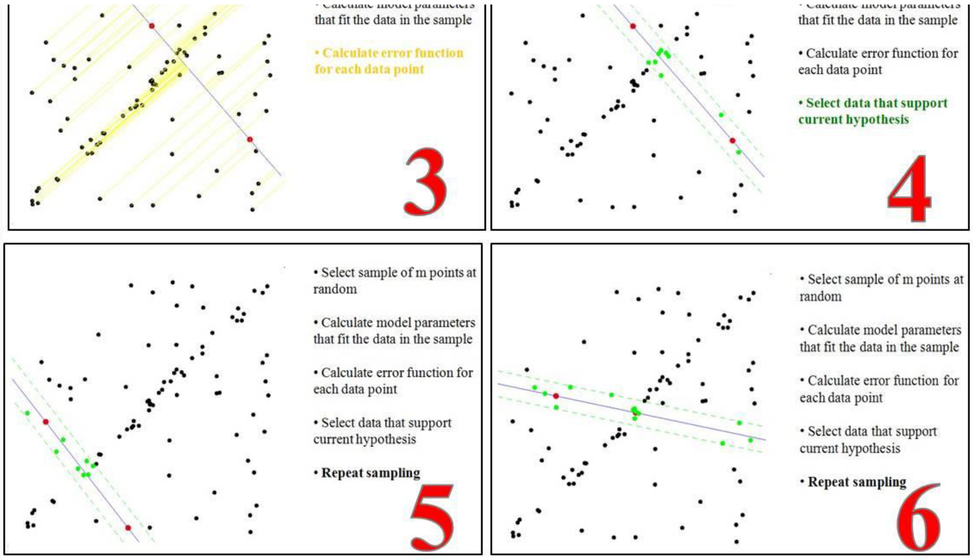
2.有一个可以解释可信观测数据的参数化模型

RANSAC的思想是（引自wiki，汉化部分有修改）：

1. Select a random subset of the original data. Call this subset the *hypothetical inliers*. 从观测数据中随机选择一个子集，称之为*hypothetical inliers*。
2. A model is fitted to the set of hypothetical inliers.估计出适合于这些个*hypothetical inliers*的模型
3. All other data are then tested against the fitted model. Those points that fit the estimated model well, according to some model-specific loss function, are considered as part of the *consensus set*. 用这个模型测试其他的数据，根据损失函数，得到符合这个模型的点，称为一致性集合：consensus set。
4. The estimated model is reasonably good if sufficiently many points have been classified as part of the consensus set. 如果足够多的数据都被归类于一致性集合，那么说明这个估计的模型是正确的；如果这个集合中的数据太少，那么说明模型不合适，弃之，返回第一步。
5. Afterwards, the model may be improved by reestimating it using all members of the consensus set.最后，根据一致性集合中的数据（可以认为是可靠的数据了），用最小二乘的方法重新估计模型。

根据上述思想，如果不停的迭代，就会得到一个最优的模型。如下图所示：





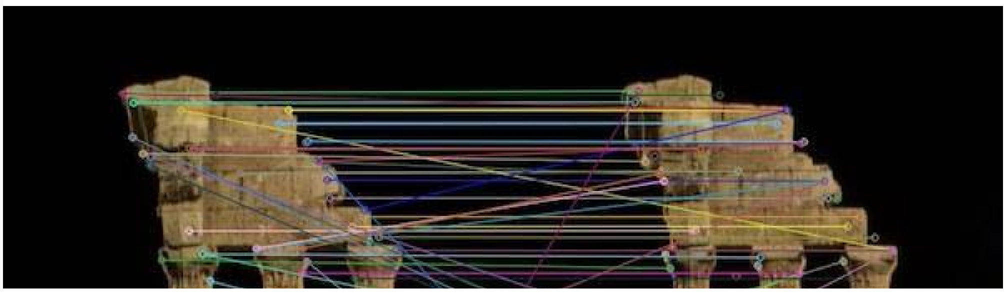
RANSAC有什么用？

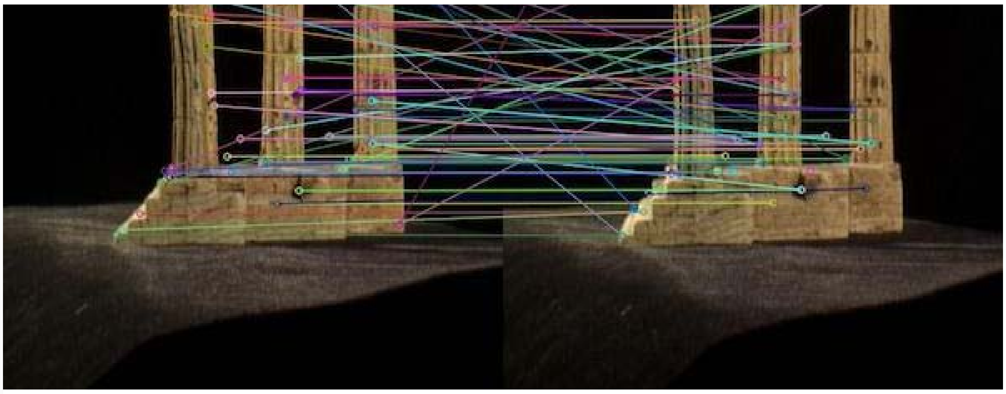
可以用于图像拼接，如果你年纪比较大，应该记得Microsoft有一款叫photosynth的产品：

怎么用RANSAC拼接呢？

图像和图像之间的关系可以用一个单应性矩阵描述，即x1=H*x2。

x1和x2就是同名点齐次坐标向量，我们可以用SIFT或SURF算子找到：





但是可以看出，虽然大部分匹配是正确的，但有一些匹配是错误，这些同名点就构成了“受污染的观测数据”，也是RANSAC的适用条件。

用RANSAC估计H的步骤如下（参考http://eric-yuan.me/ransac/）：

1. Select four feature pairs (at random) 随机找4对特征点
2. Compute homography H 计算H
3. Compute inliers where ||pi’, H pi|| < ε (if not enough times, goto 1.) 找到符合H的inliers
4. Keep largest set of inliers 直到这个H有最多的inliers
5. Re-compute least-squares H estimate using all of the inliers 用inliers和最小二乘重新估计H

同名点齐次坐标和H的关系可以写为：

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1' x_1 & -x_1' y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1' x_1 & -y_1' y_1 & -y_1' \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n' x_n & -x_n' y_n & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n' x_n & -y_n' y_n & -y_n' \end{bmatrix}}_{\substack{\mathbf{A} \\ 2n \times 9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\substack{\mathbf{h} \\ 9}} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\substack{\mathbf{0} \\ 2n}}$$

4个点对正好构成唯一解，http://eric-yuan.me/ransac/中用QR分解的方法求出H。不停迭代，直到求出最终的inliers，到第5步用最小二乘求解H时，可以用SVD分解（http://blog.csdn.net/dsbatigol/article/details/9625211）。

基于OpenCV的图像拼接的代码在这里：https://ramsrigoutham.com/tag/ransac/，随手贴上来，并感谢作者：

[cpp]

```
#include <stdio.h>
#include <iostream>

#include "opencv2/core/core.hpp"
#include "opencv2/features2d/features2d.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/nonfree/nonfree.hpp"
```

```
#include "opencv2/nonfree/nonfree.hpp"
#include "opencv2/calib3d/calib3d.hpp"
#include "opencv2/imgproc/imgproc.hpp"

using namespace cv;

void readme();

/** @function main */
int main( int argc, char** argv )
{
 if( argc != 3 )
 { readme(); return -1; }

// Load the images
 Mat image1= imread( argv[2] );
 Mat image2= imread( argv[1] );
 Mat gray_image1;
 Mat gray_image2;
 // Convert to Grayscale
 cvtColor( image1, gray_image1, CV_RGB2GRAY );
 cvtColor( image2, gray_image2, CV_RGB2GRAY );

imshow("first image",image2);
 imshow("second image",image1);

if( !gray_image1.data || !gray_image2.data )
 { std::cout<< " --(!) Error reading images " << std::endl; return -1; }

//-- Step 1: Detect the keypoints using SURF Detector
 int minHessian = 400;

SurfFeatureDetector detector( minHessian );

std::vector< KeyPoint > keypoints_object, keypoints_scene;

detector.detect( gray_image1, keypoints_object );
 detector.detect( gray_image2, keypoints_scene );

//-- Step 2: Calculate descriptors (feature vectors)
 SurfDescriptorExtractor extractor;

Mat descriptors_object, descriptors_scene;

extractor.compute( gray_image1, keypoints_object, descriptors_object );
 extractor.compute( gray_image2, keypoints_scene, descriptors_scene );

//-- Step 3: Matching descriptor vectors using FLANN matcher
 FlannBasedMatcher matcher;
 std::vector< DMatch > matches;
 matcher.match( descriptors_object, descriptors_scene, matches );

double max_dist = 0; double min_dist = 100;

//-- Quick calculation of max and min distances between keypoints
 for( int i = 0; i < descriptors_object.rows; i++ )
 { double dist = matches[i].distance;
 if( dist < min_dist ) min_dist = dist;
 if( dist > max_dist ) max_dist = dist;
```

```
  if( dist > max_dist ) max_dist = dist;
  }

 printf("-- Max dist : %f \n", max_dist );
  printf("-- Min dist : %f \n", min_dist );

 //-- Use only "good" matches (i.e. whose distance is less than 3*min_dist )
  std::vector< DMatch > good_matches;

 for( int i = 0; i < descriptors_object.rows; i++ )
  { if( matches[i].distance < 3*min_dist )
  { good_matches.push_back( matches[i]); }
  }
  std::vector< Point2f > obj;
  std::vector< Point2f > scene;

 for( int i = 0; i < good_matches.size(); i++ )
  {
  //-- Get the keypoints from the good matches
  obj.push_back( keypoints_object[ good_matches[i].queryIdx ].pt );
  scene.push_back( keypoints_scene[ good_matches[i].trainIdx ].pt );
  }

 // Find the Homography Matrix
  Mat H = findHomography( obj, scene, CV_RANSAC );
  // Use the Homography Matrix to warp the images
  cv::Mat result;
  warpPerspective(image1,result,H,cv::Size(image1.cols+image2.cols,image1.rows));
  cv::Mat half(result,cv::Rect(0,0,image2.cols,image2.rows));
  image2.copyTo(half);
  imshow( "Result", result );

  waitKey(0);
  return 0;
  }

 /** @function readme */
  void readme()
  { std::cout << " Usage: Panorama < img1 > < img2 >" << std::endl; }
```

最后提一句，RANSAC称为RANdom SAmple Consensus，即随机采样一致算法，发表于1981：

- Martin A. Fischler & Robert C. Bolles (June 1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography" (PDF).