《计算机视觉-openCV应用技术》

第六章

图像检索以及基于图像描述符的搜索

李策

中国矿业大学（北京）计算机科学与技术系

E-mail: celi@cumtb.edu.cn

提纲

第六章 图像检索以及基于图像描述符的搜索

- 6.1 特征检测算法
  - 6.1.1 特征定义
  - 6.1.2 使用 DoG 和 SIFT 提取特征及描述
  - 6.1.3 使用快速 Hessian 算法和 SURF 来提取和检测特征
  - 6.1.4 基于ORB的特征检测和特征匹配
  - 6.1.5 ORB 特征匹配
  - 6.1.6 K - 最邻近匹配
  - 6.1.7 FLANN 匹配
  - 6.1.8 FLANN 的单应性匹配
  - 6.1.9 基于文身取证的应用程序示例
- 6.2 总结

6.1 特征检测算法

OpenCV 可以检测图像的主要特征，然后提取这些特征，使其成为图像描述符，这些图像特征可作为图像搜索的数据库；此外可以利用关键点将图像拼接 stitch 起来，组成一个更大的图像。如将各照片组成一个360度的全景照片。

本章节将介绍使用 OpenCV 来检测图像特征，并利用这些特征进行图像匹配和搜索。本章节选取一些图像，检测它们的主要特征，并通过单应性（homography）来检测这些图像是否存在于另一个图像中。

6.1 特征检测算法

特征检测和提取算法有很多，OpenCV 中常用的有如下几种：

- Harris - 检测角点
- SIFT - 检测斑点（blob）
- SURF - 检测斑点
- FAST - 检测角点
- BRIEF - 检测斑点
- ORB - 该算法代表带方向的FAST算法与具有旋转不变性的BRIEF 算法

6.1 特征检测算法

通过以下方法进行特征匹配：

- 暴力（Brute-Force）匹配法
- 基于 FLANN 的匹配法

可以通过单应性进行空间验证

6.1.1 特征定义

特征就是图像有意义的区域。该区域具有独特性或易于识别性。

- 角点及高密度区域是很好的特征，大量重复的模式或低密度区域则是不好的特征；如图像中的蓝色天空就不是很好的特征。
- 边缘：将图像分为两个区域
- 斑点：与周围有很大差别的图像区域

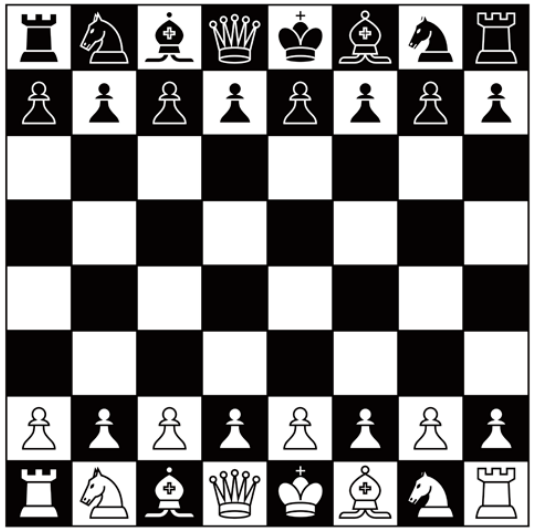
大多数特征检测算法都会涉及图像的角点、边和斑点的识别，也有一些涉及脊向（ridge）的概念，可认为脊向是细长物体的对称轴（如识别图像中的一条路）。

6.1.1 特征定义

检测角点特征：

采用 cornerHarris 检测角点特征的分析对象是国际象棋，主要是因为方格图案适合多种类型的特征检测，国际象棋很受欢迎，下图是国际象棋的示例图片

cornerHarris 函数：可以检测图像的角点



6.1.1 特征定义

```
dst = cv2.cornerHarris(gray, 2, 23, 0.04)
```

- gray - 灰度图
- 2 - 滑块窗口尺寸 / 邻域大小
- 23 - Sobel算子的中孔
- 0.04 -自由参数，经验值 0.04~0.06

6.1.1 特征定义

```
img[dst > 0.01 * dst.max()] = [0, 0, 255]
```

代码会将检测到的角点标记为红色，角点标记大小与cv2.cornerHarris() 中的第二个参数有关，参数值越小，标记角点的记号越小，下图是最终结果：

6.1.2 使用 DoG 和 SIFT 提取特征及描述

上节cv2.cornerHarris() 函数可很好地检测角点，这与角点本身的特性有关；但是该函数会因为图像的大小而产生不同的识别结果，较小的图会丢失更多的角点信息。

- SIFT：尺度不变特征变换，该函数会对不同的图像尺度（尺度不变特征变换）输出相同的结果。
- DoG：是对同一图像使用不同高斯滤波器所得到的结果,其与使用DoG 技术进行边缘检测的原理是一致的。采用DoG 操作的最终结果会得到感兴趣的区域（关键点）。

6.1.2 使用 DoG 和 SIFT 提取特征及描述

1. 使用Python的sys模块将图像路径通过命令行参数传递给脚本

```
imgpath = sys.argv[1]
img = cv2.imread(imgpath)
```

2. 创建sift对象，计算灰度图

```
sift = cv2.xfeatures2d.SIFT_create()
```

6.1.2 使用 DoG 和 SIFT 提取特征及描述

3. 在图像上绘制关键点

```
img = cv2.drawKeypoints(
    image=img,
    outImage=img,
    keypoints=keypoints,
    #该标志意味对图像上的每一个关键点都绘制了圆圈和方向
    flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS,
    color=(51, 163, 236))
```

6.1.2 使用 DoG 和 SIFT 提取特征及描述

关键点剖析：

- pt（点）属性表示图像中关键点的x坐标和y坐标
- size 属性表示特征的直径
- angle 属性表示特征的方向
- response 属性表示关键点强度
- octave 属性表示特征所在金字塔的层级
- ID 对象表示关键点的 ID

6.1.3 使用快速 Hessian 算法和 SURF 来提取和检测特征

- SIFT算法：David Lowe于1999年发表的，距现在只有22年时间

  SIFT 采用 DoG 检测关键点后提取关键点周围的特征。

- SURF特征检测算法：Herbert Bay 于2006 年发表，该算法比 SIFT 快好几倍，它吸收了 SIFT 算法的思想。

  SURF 采用快速 Hessian 算法检测关键点，并提取特征

6.1.3 使用快速 Hessian 算法和 SURF 来提取和检测特征

此外，尽管 SURF 和 SIFT 这两个特征检测算法所提供的 API 不相同，但通过简单修改前的脚本就可以动态选择特征检测算法，不必重写整个程序。具体代码如P90，运行结果：

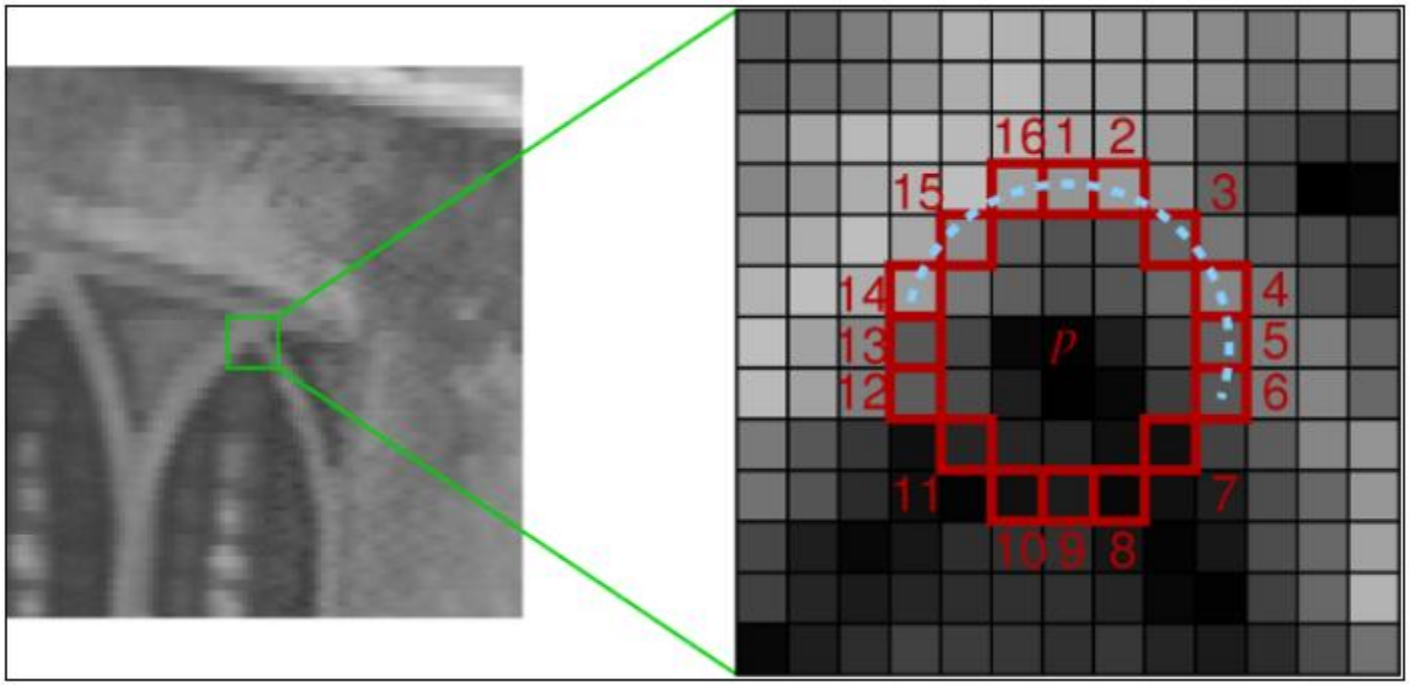
6.1.4 基于ORB的特征检测和特征匹配

ORB 将基于 FAST 关键点检测的技术和基于 BRIEF 描述符的技术相结合，因此首先学习 FAST 和 BRIEF ，然后再讨论 Brute-Force 匹配（其中的一种特征匹配算法）并展示一个特征匹配的例子。

6.1.4 基于ORB的特征检测和特征匹配

1. FAST 算法会对输入图像中的每一个像素进行计算。

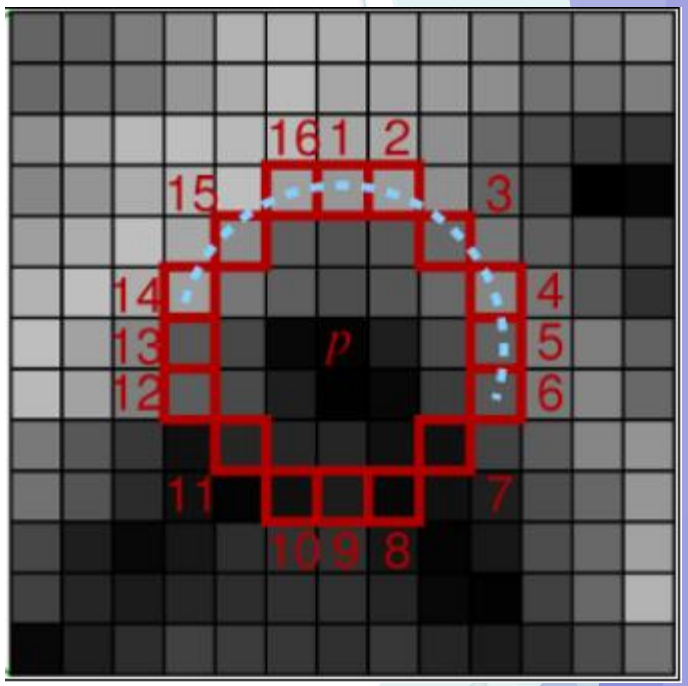
在某一像素周围绘制一个圆，比较区域圆上的点与该像素的差值。



6.1.4 基于ORB的特征检测和特征匹配

如右图，首先标记测试像素点，分别为1、5、9、13；

当四分之三的测试像素值均为 brighter（darker）时，另外四分之一的测试像素值必须为 darker（brighter），那么这个圆心就被称之为角点；如全都为 brighter 或 darker 、两个为 brighter 或 darker 时，则该圆心不是角点。



6.1.4 基于ORB的特征检测和特征匹配

2. BRIEF 是一个描述符，而不是一种算法。

SIFT 和 SURF 分析图像时，核心函数为 detectAndCompute 函数，该函数通过检测和计算返回两个结果；检测结果是一组关键点，计算结果是描述符。

关键点描述符是图像的一种表示，因此可比较两个图像的关键点描述符；并找到它们的共同之处，所以描述符可作为特征匹配的一种方法（gateway）

BRIEF 是目前最快的描述符，其理论相当复杂，但 BRIEF 采用了一系列的优化措施，使其成为不错的特征匹配方法。

6.1.4 基于ORB的特征检测和特征匹配

3. 暴力匹配

暴力匹配方法是一种描述符匹配方法，该方法会比较两个描述符，并产生匹配结果的列表。

第一个描述符的所有特征都用来和第二个描述符的特征进行比较，每次比较都会给出一个距离值，其中比较结果中最好的那个被认为是一个匹配。

暴力往往与穷举所有可能的组合（穷举可能字符进行密码破解）有关。

6.1.5 ORB 特征匹配

在 ORB 的论文中，作者得到如下结果：

- 向 FAST 增加一个快速、准确的方向分量（component）
- 能高效计算带方向的 BRIEF 特征
- 基于带方向的 BRIEF 特征的方差分析和相关性分析
- 在旋转不变性条件下学习一种不相关的 BRIEF 特征，这会在最邻近的应用中得到较好的性能。

6.1.5 ORB 特征匹配

A. 首先加载两幅图（查询图像和训练图像）

```
img1 = cv2.imread('./cat.jpg')
img2 = cv2.imread('./cat_rabbit.jpg')
```

B. 创建ORB特征检测器和描述符

```
orb = cv2.ORB_create()
kp1, des1 = orb.detectAndCompute(img1, None)
kp2, des2 = orb.detectAndCompute(img2, None)
```

6.1.5 ORB 特征匹配

C. 对查询图像和训练图像都要检测，然后计算关键点和描述符，BFMatcher 创建匹配器对象，

```
bf =
cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
```

D. match 实现暴力匹配

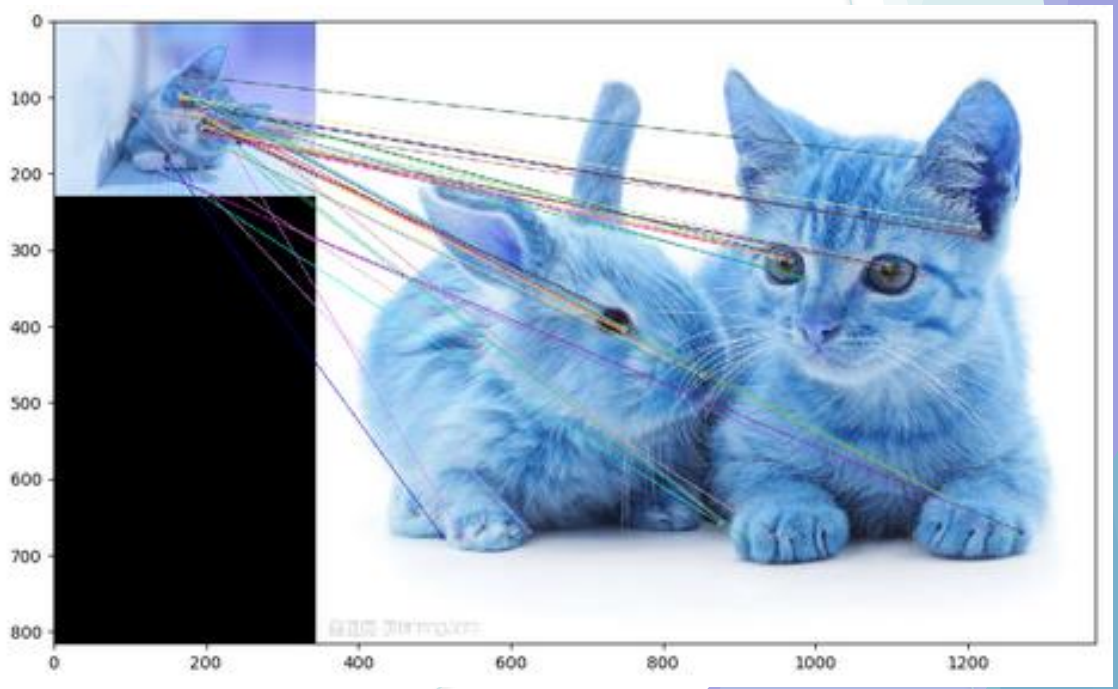
```
matches = bf.match(des1, des2)
matches = sorted(matches, key=lambda x: x.distance)
```

6.1.5 ORB 特征匹配

E. 现已经获取所有需要的信息，将其图标来绘制这些匹配

```
img3 = cv2.drawMatches(img1, kp1, img2, kp2,
matches[:40], img2, flags=2)
plt.imshow(img3)
plt.show()
```

运行结果：



6.1.6 K - 最邻近匹配

有很多可以用来检测匹配的算法，从而可以绘制这些匹配。k - 最邻近（KNN）是其中一种匹配检测算法；在所有的机器学习的算法中，KNN 可能是最简单的。

在脚本中使用 KNN ，只需要上节 ORB 特征匹配进行修改即可：

- KNN 代替暴力匹配
- 对应匹配函数替换 drawMatches -> drawMatchesKnn

6.1.6 K - 最邻近匹配

A. 首先加载两幅图（查询图像和训练图像）

```
img1 = cv2.imread('./cat.jpg', 0)
img2 = cv2.imread('./cat_rabbit.jpg', 0)
```

B. 创建ORB特征检测器和描述符

```
orb = cv2.ORB_create()
kp1, des1 = orb.detectAndCompute(img1, None)
kp2, des2 = orb.detectAndCompute(img2, None)
```

6.1.6 K - 最邻近匹配

C. 对查询图像和训练图像都要检测，计算关键点和描述符

```
bf =
cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
```

D. knn 匹配

```
matches = bf.knnMatch(des1, des2, k=1)
```

E. 现已经获取所有需要的信息，将其图标来绘制这些匹配

```
img3 =
cv2.drawMatchesKnn(img1, kp1, img2, kp2, matches,
img2, flags=2)
```

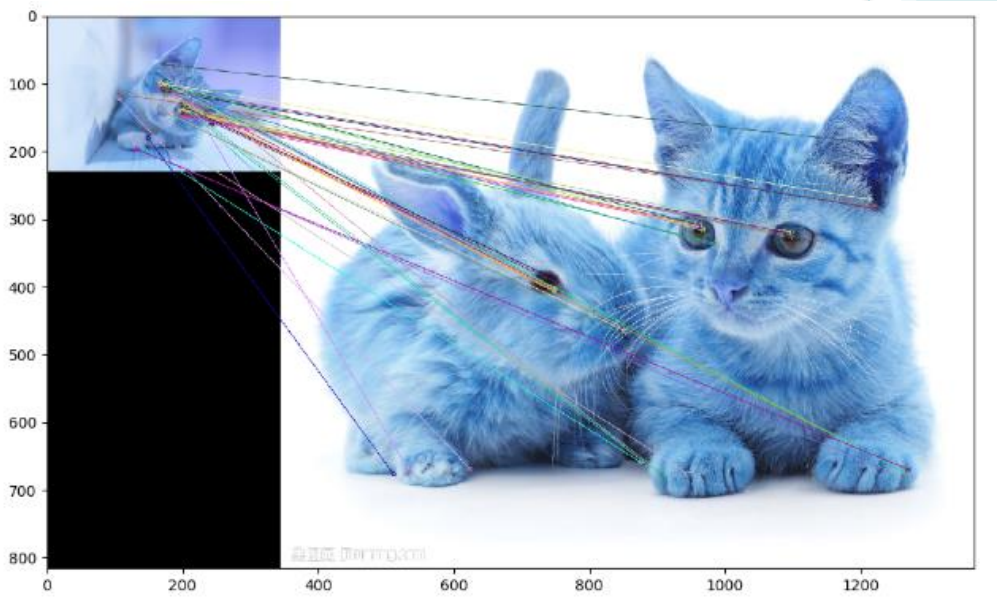
6.1.6 K - 最邻近匹配

最后的结果与ORB 运算结果类似。

match 函数与 knnMatch 函数的区别：

- match 函数返回最佳匹配
- KNN 函数返回 k 个匹配，开发人员可以用knnMatch 进一步处理这些匹配项

运行结果：



6.1.7 FLANN 匹配

FLANN（Fast Library for Approximate Nearest Neighbors）称之为近似最近邻的快速库。

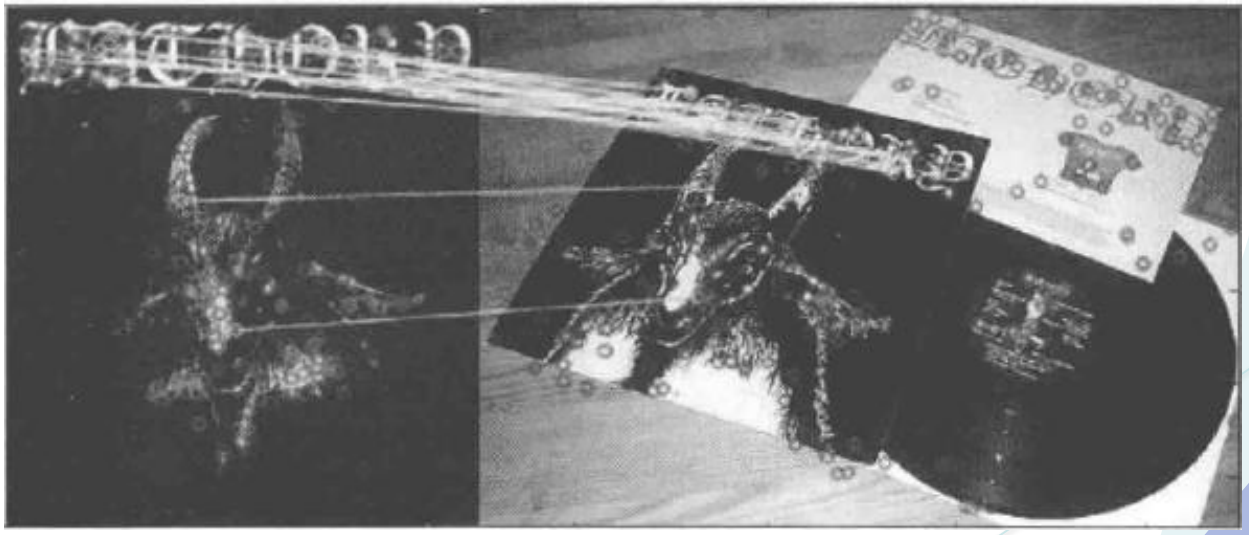
实际上，FLANN 具有一种内部机制，该机制可以根据数据本身选择最合适的算法来处理数据集，经验证，FLANN 比其它的最近邻搜索软件快 10 倍。

基于 FLANN 进行特征匹配的例子见P97

6.1.7 FLANN 匹配

脚本图像

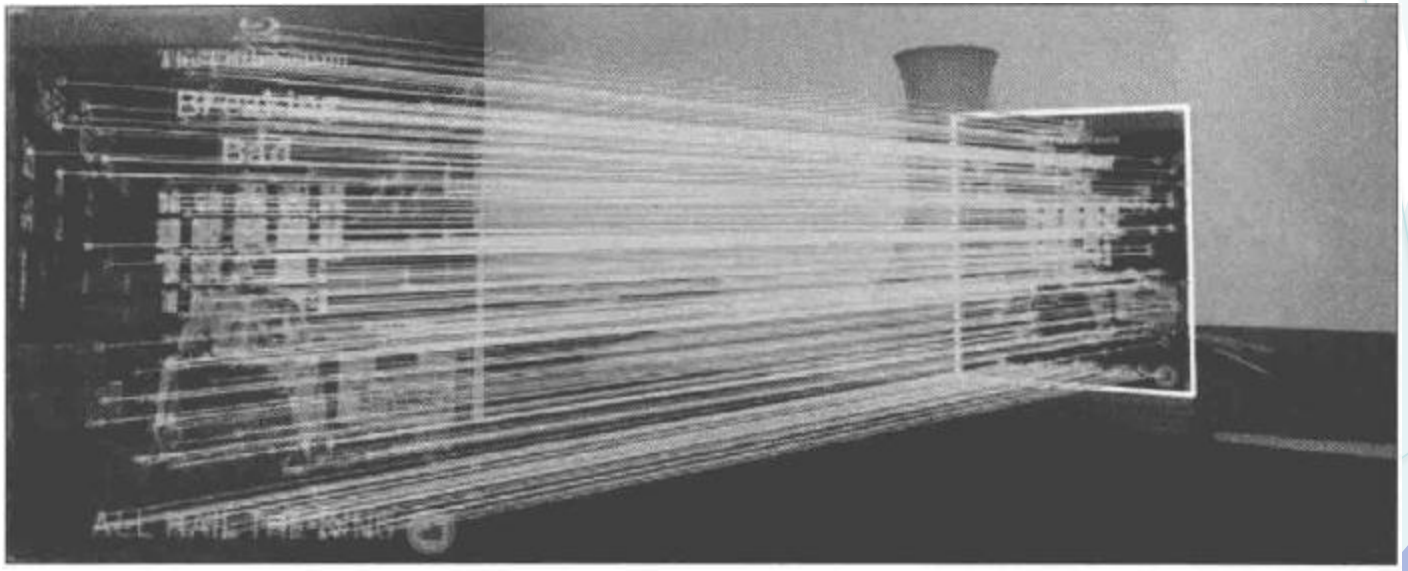
训练图像



FLANN 处理得到的结果

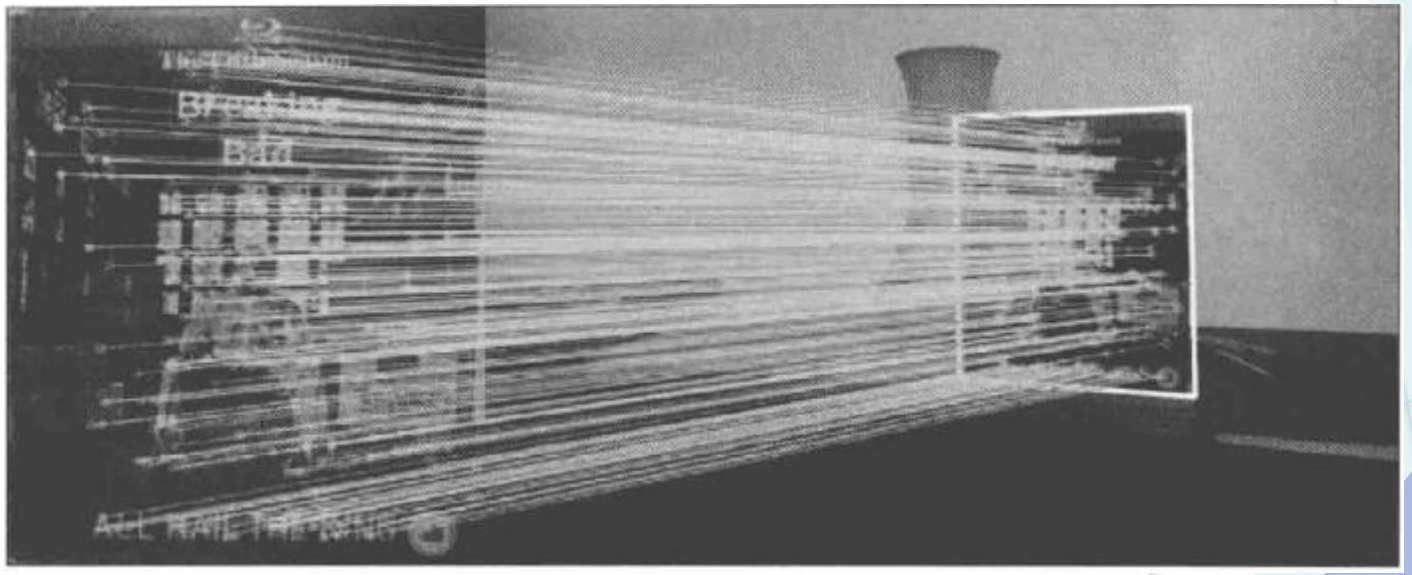
6.1.8 FLANN 的单应性匹配

单应性 - 单应性是一个条件，一幅图通过该条件出现投影畸变（perspective distortion）时，另一幅图能够与之匹配



6.1.8 FLANN 的单应性匹配

从下图可以看到：左边有一幅图像，该图像正确识别了右侧的图像，画出了关键点之间的匹配线段，而且画了一个白色边框，用来展示图像原目标在右侧发生投影畸变的效果



6.1.9 基于文身取证的应用程序示例

现有犯罪嫌疑人在罪案现场遗留下的文身原始照片/ 图案，但是尚不知犯罪嫌疑人的其他信息；已知已经建立文身数据库，现需要通过文身照片比对照出犯罪嫌疑人信息。

将上述语言分析为计算机语言，可分为两部分：

- 第一步，将图像描述符保存到文件中
- 第二步，以该照片作为查询图像，在数据库中检索匹配图像

6.1.9 基于文身取证的应用程序示例

1. 将数据库图像描述符保存到文件中

当两幅图像进行匹配和单应性分析时，不用每次都重建描述符。

编写应用程序，将数据库图像保存到文件夹中，并创建相应的描述符文件，可供后面被搜索时使用。

主要实现的过程是：

A. 加载图像

B. 创建特征检测器

C. 检测并计算

6.1.9 基于文身取证的应用程序示例

2. 扫描匹配

将描述符保存到文件后，接下来需要对所有描述符进行单应性处理，由此找到可能与查询图像匹配的图像。

实现步骤如下：

A. 加载文身数据库的描述符

B. 加载罪犯文身的描述符

C. FLANN 单应性匹配

D. 输出操作

6.2 总结

本章介绍了如何检测图像特征以及如何为描述符提供特征，探讨了如何通过OpenCV提供的算法完成这个任务，介绍了实际场景的应用，从而理解了这些概念在真实世界中的应用。

已经熟悉检测一幅图像（或视频帧）特征的概念，为学习下一章打下了良好的基础。