6. SIFT

- 图像关键点（keypoint）的提取
- SIFT 描述子的计算
- 图像特征匹配

SIFT 简介

- SIFT 算法是 David Lowe 于1999年提出的局部特征描述子，并于2004年进行了更深入的发展和完善。SIFT 特征匹配算法可以处理两幅图像之间发生平移、旋转、仿射变换情况下的匹配问题，具有很强的匹配能力。
- Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*
- 总体来说，SIFT 算子具有以下特性：

1. SIFT 特征是图像的局部特征，对平移、旋转、尺度缩放、亮度变化、遮挡和噪声等具有良好的不变性，对视觉变化、仿射变换也保持一定程度的稳定性。
2. 独特性好，信息量丰富，适用于在海量特征数据库中进行快速、准确的匹配。
3. 多量性，即使少数的几个物体也可以产生大量 SIFT 特征向量。
4. 速度相对较快，经优化的 SIFT 匹配算法甚至可以达到实时的要求。
5. 可扩展性强，可以很方便的与其他形式的特征向量进行联合。

SIFT 简介

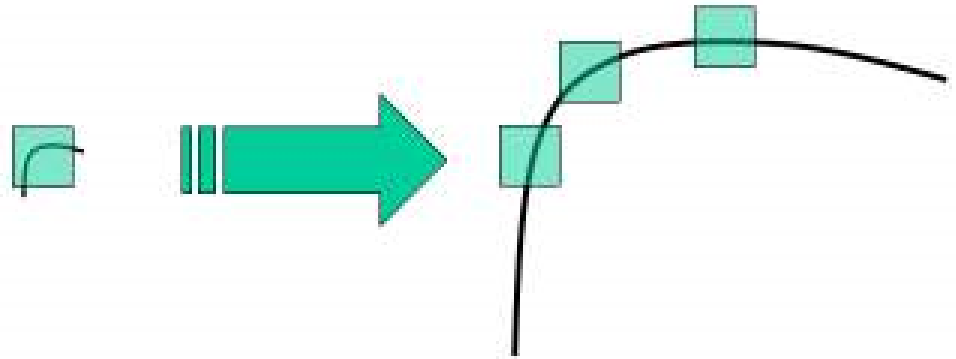
- SIFT 算法的实质是在不同的尺度空间上查找关键点（特征点），并计算出关键点的方向。SIFT 所查找到的关键点是一些十分突出，不会因光照，仿射变换和噪音等因素而变化的点，如角点、边缘点、暗区的亮点及亮区的暗点等。

- 主要步骤：

1. 检测尺度空间极值点
2. 精确定位特征点
3. 为每个特征点指定方向参数
4. 生成特征点描述子
5. 匹配特征点

尺度空间极值检测

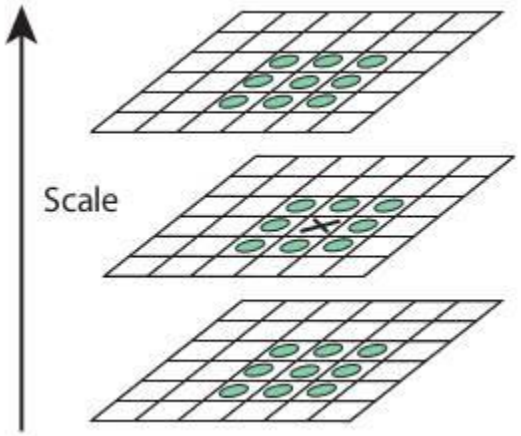
- 一些角点检测器具有旋转不变性，如 Harris Corner Detector。这意味着即使图像旋转，我们也可以找到相同的角点，因为显然角点在旋转后的图像中也是角点。但是如果图像缩放，角点可能就不再是角点。以下图为例，在小图像中使用一个小窗口能够检测出角点，然而将图像放大后，在同一窗口中的图像变得平坦，无法检测出角点。所以 Harris 角点不是尺度不变的。

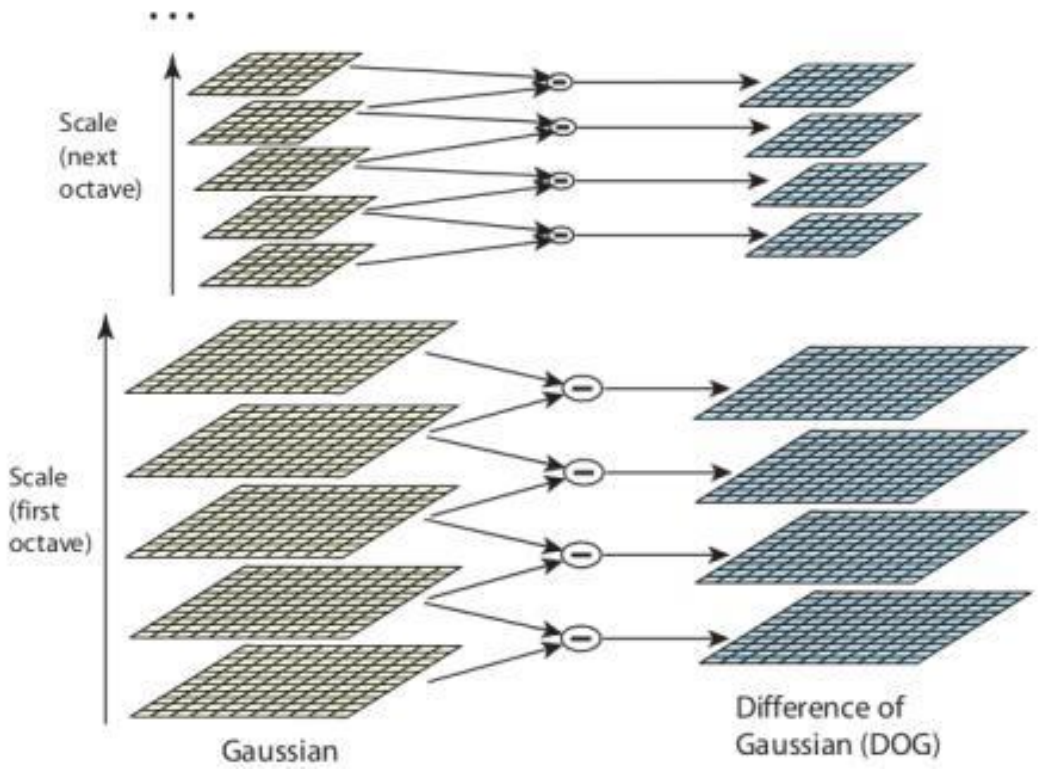


- 从上图可以看出，我们不能使用同一个窗口来检测不同尺度空间中的角点。检测小的角点可以用小的窗口，但检测更大的角点则需要更大的窗口，为此需要进行尺度空间滤波。使用不同 σ 值的高斯拉普拉斯算子（LoG）对图像进行卷积。具有不同 σ 值的 LoG 可以检测不同大小的斑点。简而言之，σ 相当于一个尺度变换因子。
- 例如，在上图中，使用具有低 σ 的高斯核可以检测出小的角点，而具有高 σ 的高斯核则适合于检测大的角点。因此，我们可以在尺度空间和二维平面中找到局部最大值（x，y，σ），这意味着在 σ 尺度中的（x，y）处有一个潜在的特征点。

尺度空间极值检测

- 但是 LoG 的计算量较大，因此 SIFT 算法使用高斯差分近似计算 LoG。分别使用方差值为 σ 和 kσ 的高斯核对图像进行模糊处理，二者的差值就是 DoG。
- 算出 DoG 后，搜索不同尺度空间和二维平面中的局部最大值。例如，可以将图像中的一个像素与其 8 个相邻像素、尺度空间上一层中相邻的 9 个像素以及尺度空间下一层中相邻的 9 个像素做比较。如果它是一个局部最大值，那么它就是一个潜在的特征点。这几乎意味着特征点是相应尺度空间的最佳代表。





尺度空间极值检测

- 高斯金字塔的组内尺度与组间尺度：$o(s)=\sigma_o \cdot 2^{s/S}$
- 组内尺度是指同一组（octave）内，不同层（level）之间的尺度关系。组内相邻层间尺度可以简化为：

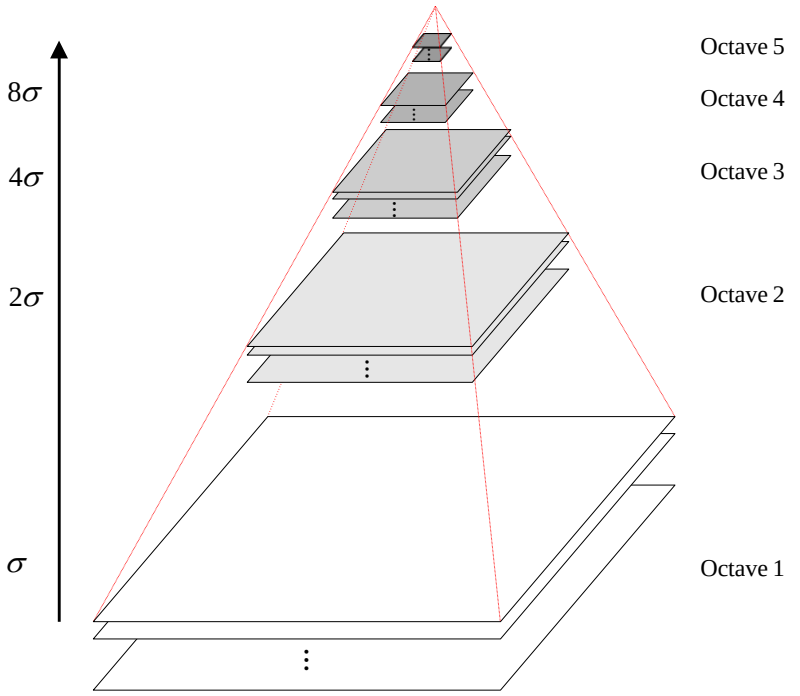
$$\sigma_{s+1}=\sigma_s \cdot 2^{1/S}$$

- 组间尺度是指不同组之间的尺度关系，相邻组的尺度可以简化为：

$$\sigma_{o+1}(s)=\sigma_o \cdot 2^{s+S/S}$$

$$\sigma_o \cdot 2^{s+S/S}=2\sigma_o \cdot 2^{s/S}$$

- 由此可见，相邻两组的同一层尺度为2倍的关系。
- 对于参数的选取，该论文给出了一些经验数据，即：建立 4 组不同尺寸的尺度空间，对于每一组尺度空间使用不同的 σ 构建 5 层图像。其中，σ 的初始值为 1.6，k 的初始值为 $\sqrt{2}$。



特征点精确定位

- 在找到潜在的特征点后，则需要对其进行细化以获得更准确的结果。先对该点的尺度空间进行泰勒展开

$$D(X)=D+\frac{\partial D^T}{\partial X}X+\frac{1}{2}X^T\frac{\partial^2 D}{\partial X^2}X$$

令该函数的导数值为零，即

$$\hat{X}=-\frac{\partial D^T}{\partial X}(\frac{\partial^2 D}{\partial X^2})^{-1}$$

- 来获得更准确的极值位置 $\hat{X}=(x,y,\sigma)^T$ 。令

$$D(X)=D+\frac{1}{2}\frac{\partial D^T}{\partial X}X$$

- 并且如果此极值处的强度小于阈值，则将其忽略。论文中使用的阈值为 0.03。
- DoG 对边缘更加敏感，因此需要去除边缘。为此可以使用类似于 Harris 角点检测的思路。论文作者使用 2x2 Hessian 矩阵

$$H=\begin{pmatrix}D_{xx} & D_{xy}\\ D_{xy} & D_{yy}\end{pmatrix}$$

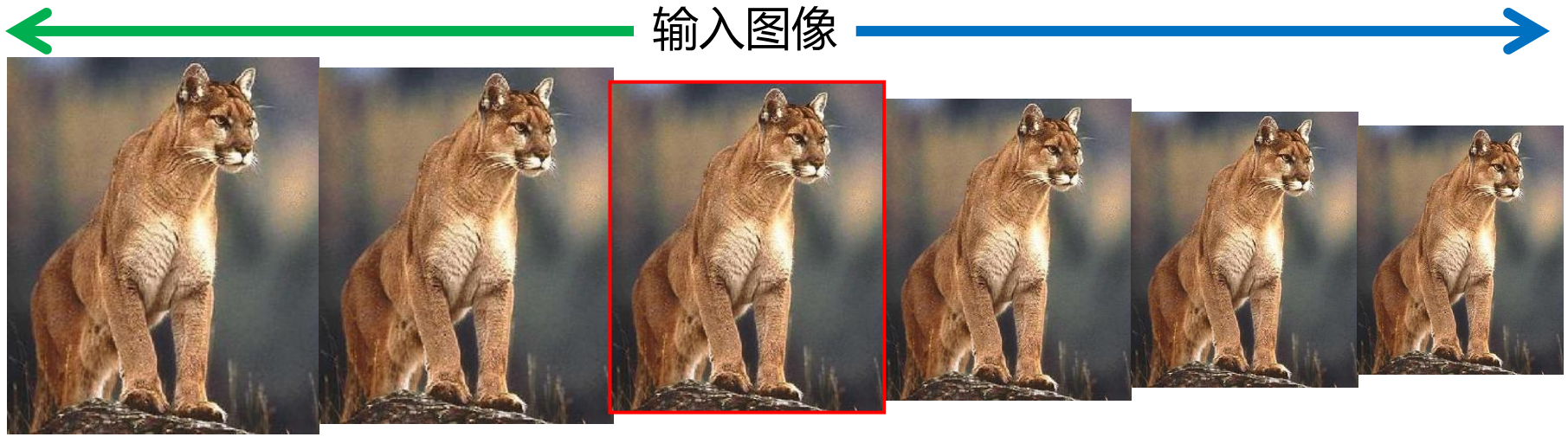
来计算主曲率。

- 当该矩阵的一个特征值大于另一个特征值时，则检测到了边界。论文作者使用了一个简单的函数对此进行判断，若两特征值的比率大于阈值，则丢弃该特征点。论文中给出的边界阈值是 10。
- 在将所有低对比度的特征点和边缘特征点去除后，剩下的就是我们所感兴趣的特征点。

特征点提取的简化操作

- 上述在尺度空间中提取极值点再精确定位和消除边缘响应的方法实现起来较为复杂，实验中可用另一种较简单的替代方案，即使用 Harris 角点提取函数在图像金字塔中提取角点。
- 与用高斯差分构建的尺度空间不同，图像金字塔通过直接缩放图像的方式构建尺度空间。

```
# OpenCV 中改变图像尺寸
im2 = cv2.resize(im1, (size, size))
```

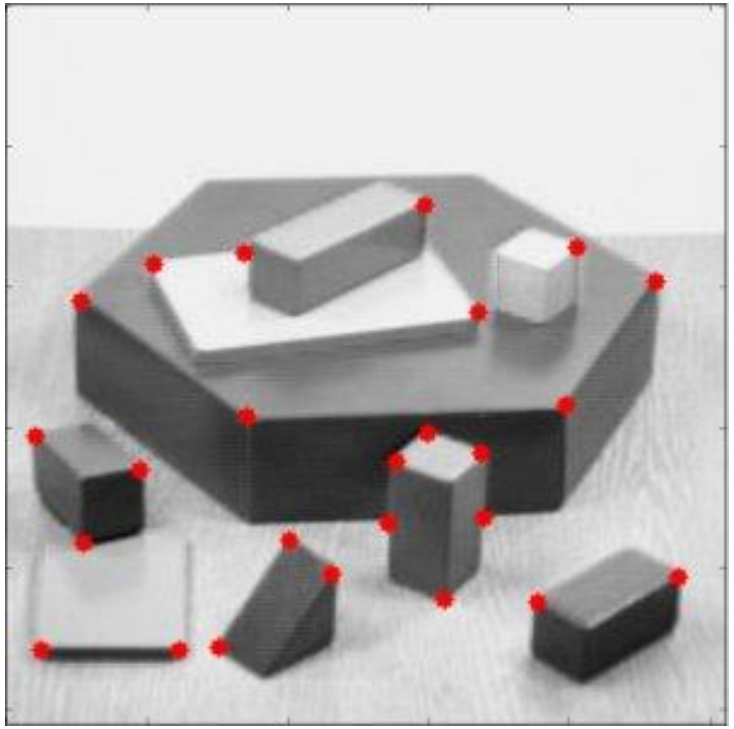


图像金字塔

特征点提取的简化操作

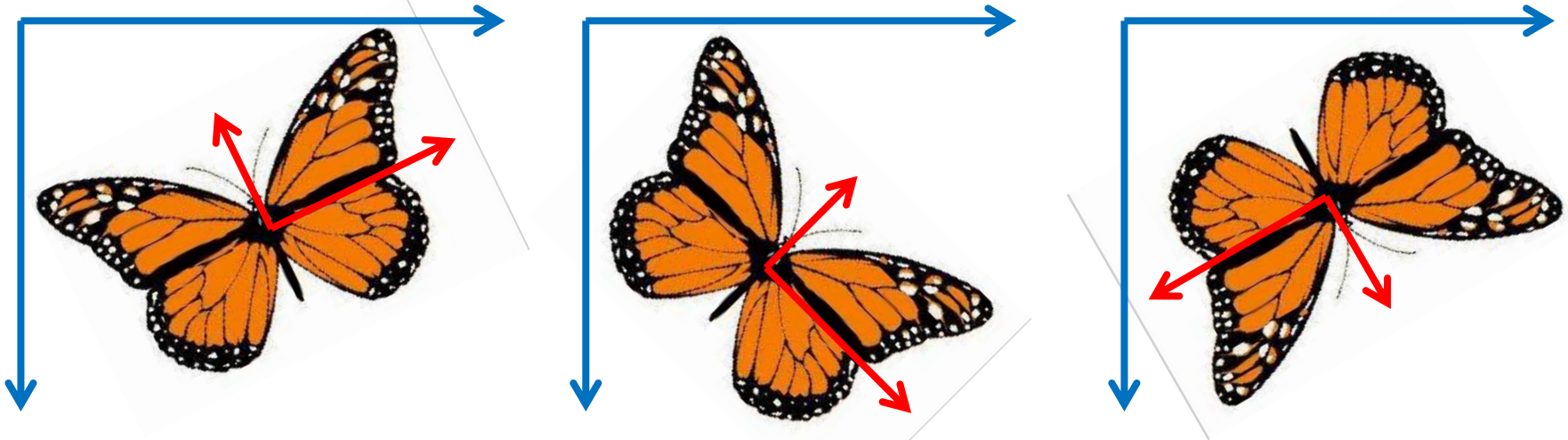
- OpenCV 中也实现了角点检测函数 goodFeaturesToTrack，可用于简化特征点提取，在实验中可直接调用。
- Jianbo Shi, C. Toamsi. Good Features to Track. IEEE Conference on Computer Vision and Pattern Recognition, 1994.

```
# 使用 goodFeaturesToTrack 函数计算特征点

cv2.goodFeaturesToTrack(image, maxCorners,
qualityLevel, minDistance[, corners[, mask
[, blockSize[, useHarrisDetector[, k]]]]])

# 经验参数：qualityLevel = 0.01, minDistance = 10,
blockSize = 3, k = 0.04
```



SIFT 描述子——方向参数分配

- 如下图所示，蓝色箭头表示图像坐标系，红色箭头表示物体坐标系。图像坐标系即观测图像的横向和纵向两个方向，也是计算机处理图像时所参照的坐标系。而物体参照系是人在认知物体时参照的坐标系。人脑之所以能够分辨不同方向的同一物体，是因为能够从物体的局部细节潜在地建立起物体坐标系，并建立其和图像坐标系之间的映射关系。



- 为了实现图像的旋转不变性，需要为每个特征点指定物体坐标系的主方向，即以关键点为中心的邻域内的主要梯度方向。该坐标系是由关键点及其周围像素本身的性质定义的，因此具有旋转不变性。

SIFT 描述子——方向参数分配

- 假设某尺度下某特征点为 L(x, y)，对于其 m × m (m = 1.5σ) 邻域的每个点，计算其在该尺度下的梯度强度和方向

$$m(x,y)=\sqrt{\left(L(x+1,y)-L(x-1,y)\right)^2+\left(L(x,y+1)-L(x,y-1)\right)^2}$$

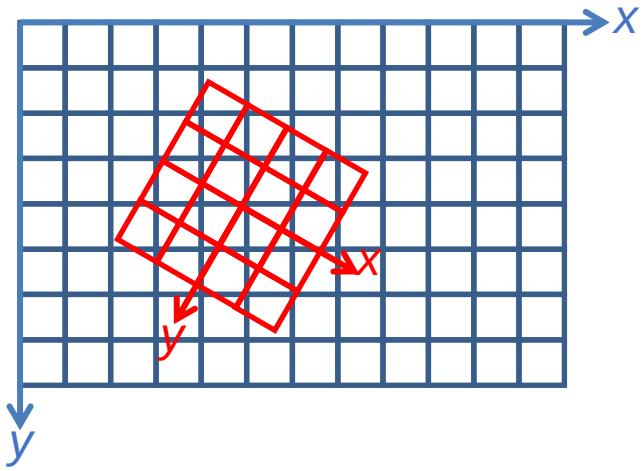
$$\theta(x,y)=\tan^{-1}\left(\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right)$$

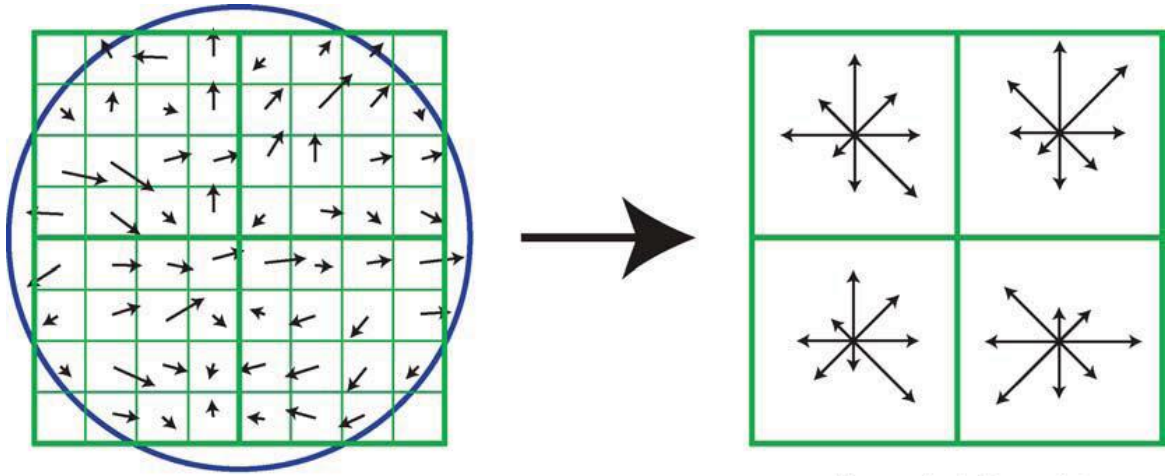
- 其中，梯度的角度值取 0° - 360°。
- 将梯度方向 360° 平均分成 36 bins（10° 一个 bin），并将之前计算出的每个像素的梯度以角度 θ(x, y) 和强度 m(x, y) 进行评分，选出关键点的主方向。该方向即为该特征点的物体坐标系方向。（本实验仅需考虑主方向）。

(a) (b)

SIFT 描述子——描述子的计算

- 考虑物体坐标系中，相对特征点周围的 16 × 16 个像素。将这 256 个像素分成 4 × 4 组，其中每组包含 4 × 4 个像素点。对每一组中像素点的梯度，按照此前计算主方向的方式对其直方图进行统计，并拼接起来，即得到了特征点的 SIFT 描述子。
- 此处采用的直方图为 8 bins（45° 一个 bin），而非此前的 36 bins。对于每个关键点，生成的 SIFT 描述子维度为 8 × 4 × 4 = 128。再将这些描述子左右拼接，则得到了 *Number_of_Keypoints* × 128 的矩阵，即为该图片的 SIFT 描述子。





SIFT 描述子——描述子的计算

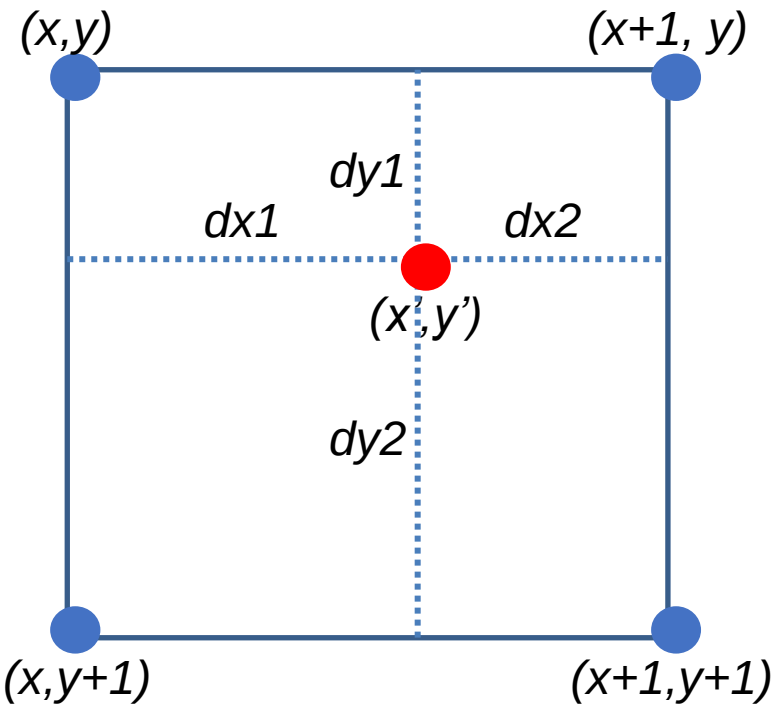
- 物体坐标系上的每一个整数点对应的图像坐标系可能不是整数，可采用最邻近插值，即图像坐标系上和它最接近的一个点：

$$\theta(x', y') = \theta(\text{Round}(x'), \text{Round}(y'))$$

- 或更精确地，也可采用双线性插值：

$$\begin{aligned}\theta(x', y') = {} & \theta(x, y) * dx2 * dy2 \\ & + \theta(x+1, y) * dx1 * dy2 \\ & + \theta(x, y+1) * dx2 * dy1 \\ & + \theta(x+1, y+1) * dx1 * dy1\end{aligned}$$

- 其插值的意义在于，周围的四个点的值都对目标点有贡献，贡献大小与距离成正比。



特征点的匹配

- 通过对两个 SIFT 描述子做内积，我们就可以得到两个 SIFT 描述子的相似度，即

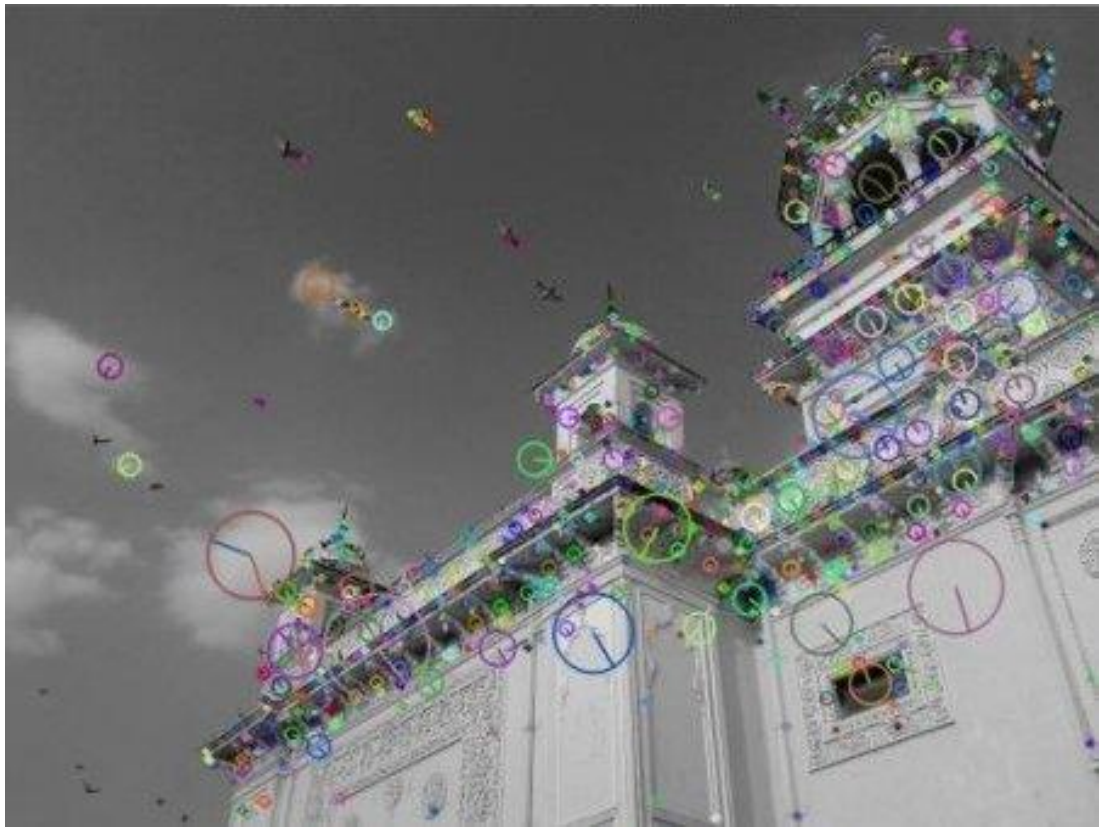
$$s(f_1, f_2) = f_1 \cdot f_2$$

- OpenCV 中实现了 Brute-Force Matcher。该匹配器计算第一组中某个特征点描述子和第二组中所有其他特征点的距离，并返回距离最近的一个特征点，认为其是一组好的匹配。对于 SIFT 描述子的匹配，可以使用 L2-Norm（欧氏距离）。

```
# BFMatcher with default params
bf = cv.BFMatcher()
matches = bf.knnMatch(des1, des2, k=2)
```

OpenCV 实现的 SIFT 函数

```
# 实例化 SIFT 函数
sift = cv2.SIFT_create()

# 计算灰度图中的特征点及 SIFT 描述子
kp, des = sift.detectAndCompute(gray, None)

# 将特征点描绘到图像上
img = cv2.drawKeypoints(gray, kp, img)

# 将特征点的大小及方向描绘到图像上
img = cv2.drawKeypoints(gray, kp, img, flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
```



作业练习

- 请在 dataset 文件夹中的所有图片中搜索 target.jpg 图片所示物体，并绘制程序认为的好的匹配。与 OpenCV 实现的 SIFT 函数比较。

- 注：**高斯差分金字塔**和**特征点提取**可用简化方法，**SIFT 描述子**需要自己计算。

- 报告中应包含实验原理，算法，流程，结果，最好有自己的心得体会和讨论，源程序附在报告最后。

参考资料

- **SIFT 算法详解** https://blog.csdn.net/zddblog/article/details/7521424
- **OpenCV SIFT 简介** https://opencv.apachecn.org/#/docs/4.0.0/5.4-tutorial_py_sift_intro
- **Shi-Tomasi 角点检测** https://opencv.apachecn.org/#/docs/4.0.0/5.3-tutorial_py_shi_tomasi
- **Feature Matching** https://opencv.apachecn.org/#/docs/4.0.0/5.9-tutorial_py_matcher