**《计算机视觉》实验报告**

**姓名：汪江豪 学号：22121630**

实验八

1. **任务1**

采用SIFT特征实现图像检索功能，即输入一张图片，在数据集中检索出相似的图片，数据集自选。

（1）基于词袋模型实现

（2）检索结果按照相似度进行排序

**a)核心代码：**

**· SIFT特征：**SIFT(Scale-Invariant Feature Transform，尺度不变特征转换)是一种图像局部特征提取算法。能够从图像中检测出具有尺度不变性和旋转不变性的关键点，并为每个管家点生成一个描述符，用于描述该区域的局部纹理信息。广泛用于图像匹配、目标识别、图像检索、三维重建等领域。

**· 词袋模型(Bag of Visual Words,BoVW)**是受自然语言处理中词袋概念启发而提出的图像表示方法。它将图像中的局部特征(如SIFT)聚类成视觉词汇，从而构建一个长度固定的特征向量(即直方图)，用于图像分类或检索任务。

**·BoVW的核心思想：**

1)特征提取：使用SIFT提取所有训练图像中的局部特征。

2)聚类生成视觉词典：通过K-Means等聚类算法将这些特征聚类成若干簇，每个簇中心作为一个视觉单词，形成视觉词典。

3)图像表示：对于每张图像，统计其特征点落在各个视觉词上的频率，形成一个直方图作为该图像的全局表示。

4)相似度计算：使用余弦相似度、欧氏距离等方法比较查询图像与数据集中图像之间的直方图差异，进行全局排序和检索。

1. 提取SIFT特征：

    def *extract\_sift\_features*(*self*, *image\_path*):

        """从图像中提取SIFT特征"""

        img = cv2.*imread*(*image\_path*, cv2.IMREAD\_GRAYSCALE)

        if img is None:

            return np.*array*([])

*# 调整图像大小，保持特征一致性*

        img = cv2.*resize*(img, (0,0), *fx*=0.8, *fy*=0.8)

*# 使用CLAHE进行对比度增强，改善特征提取*

        clahe = cv2.*createCLAHE*(*clipLimit*=2.0, *tileGridSize*=(8, 8))

        img = clahe.*apply*(img)

*# 初始化SIFT检测器*

        sift = cv2.SIFT\_create(*nfeatures*=500)

*# 检测关键点和计算描述符*

        keypoints, descriptors = sift.detectAndCompute(img, None)

        if descriptors is None:

            return np.*array*([])

        return descriptors

2.提取HOG特征：

    def *extract\_hog\_features*(*self*, *image\_path*):

        """从图像中提取HOG特征"""

        img = cv2.*imread*(*image\_path*, cv2.IMREAD\_GRAYSCALE)

        if img is None:

            return np.*array*([])

*# 调整图像大小，保持特征一致性*

        img = cv2.*resize*(img, (128, 128))

*# 计算HOG特征*

        hog = cv2.*HOGDescriptor*()

        descriptors = hog.*compute*(img)

3.构建视觉词典：

    def *build\_vocabulary*(*self*, *dataset\_path*):

        """构建视觉词典"""

        all\_features = []

*# 获取所有图像路径*

        for root, \_, files in os.*walk*(*dataset\_path*):

            for file in files:

                if file.lower().endswith(('.png', '.jpg', '.jpeg')):

                    image\_path = os.path.*join*(root, file)

*self*.image\_paths.*append*(image\_path)

*# 从每个图像中提取特征*

*print*('正在提取SIFT和HOG特征...')

        for image\_path *in* *tqdm*(*self*.image\_paths):

            descriptors = *self*.*extract\_sift\_features*(image\_path)

            if descriptors.*shape*[*0*] > 0:

*self*.*features\_dict*[*image\_path*] = descriptors

                all\_features.*append*(descriptors)

*# 提取并存储HOG特征*

            hog\_descriptor = *self*.*extract\_hog\_features*(image\_path)

            if hog\_descriptor.*shape*[*0*] > 0:

*self*.*hog\_features*[*image\_path*] = hog\_descriptor

*# 将所有特征连接成一个大矩阵*

        all\_features = np.*vstack*(all\_features)

*print*(*f*"总共从{*len*(*self*.features\_dict)}张图像中提取了{all\_features.*shape*[*0*]}个特征点")

*# 使用K-means聚类来创建视觉词汇*

*print*(*f*"正在构建包含{*self*.vocab\_size}个视觉词汇的词袋模型...")

*self*.kmeans = *KMeans*(*n\_clusters*=*self*.vocab\_size, *init*='k-means++', *random\_state*=42, *n\_init*=1, *verbose*=1)

*# 强制采样，加快KMeans收敛速度*

        max\_samples = 50000

*# 如果特征数量过多，可以随机抽样减少计算量*

        if all\_features.*shape*[*0*] > max\_samples:

            idx = np.random.choice(all\_features.*shape*[*0*], max\_samples, *replace*=False)

            sampled\_features = *all\_features*[*idx*]

*print*(*f*'强制采样后特征点数：{sampled\_features.*shape*[*0*]}')

        else:

            sampled\_features = all\_features

*# 始终使用采样后的特征点进行聚类*

*self*.kmeans.*fit*(sampled\_features)

*print*("词典构建完成！")

4. 创建视觉词汇直方图：

    def *create\_histogram*(*self*, *descriptors*):

        """创建视觉词汇直方图"""

        if *descriptors*.*shape*[*0*] == 0:

            return np.*zeros*(*self*.vocab\_size)

*# 预测每个特征点属于哪个聚类（视觉词）*

        predictions = *self*.kmeans.*predict*(*descriptors*)

        histogram = np.*zeros*(*self*.vocab\_size)

*# 计算Term Frequency(TF)*

        for prediction in predictions:

*histogram*[*prediction*] *+=* 1

*# 归一化TF*

        if np.*sum*(histogram) *>* 0:

            histogram = histogram */* np.*sum*(histogram)

*# 应用IDF权重（只有在所有直方图都已生成后才有意义）*

        if *len*(*self*.histograms) > 0:

            all\_hist = np.*array*(*list*(*self*.histograms.*values*()))

            df = np.*sum*(all\_hist *>* 0, *axis*=0)

            idf\_weights = np.log((*len*(*self*.image\_paths) + 1) / (df + 1))

            histogram = histogram \* idf\_weights

*# 否则不加IDF*

*# L2归一化*

        norm = np.linalg.*norm*(histogram)

        if norm *>* 0:

            histogram = histogram */* norm

        return histogram

5. 检索相似图片：

    def *search\_image*(*self*, *query\_image\_path*, *top\_n*=5):

        """根据查询图像检索相似图像"""

*# 提取查询图像的SIFT和HOG特征*

        query\_sift\_descriptor = *self*.*extract\_sift\_features*(*query\_image\_path*)

        query\_hog\_descriptor = *self*.*extract\_hog\_features*(*query\_image\_path*)

        if query\_sift\_descriptor.*shape*[*0*] == 0 or query\_hog\_descriptor.*shape*[*0*] == 0:

*print*("无法从查询图像中提取特征")

            return []

*# 为查询图像创建直方图(SIFT)*

        query\_histogram = *self*.*create\_histogram*(query\_sift\_descriptor)

*# HOG直接作为特征向量*

        query\_hog\_vector = query\_hog\_descriptor.*reshape*(1, -1)

*# 计算查询直方图与所有图像直方图之间的相似度*

        similarities = {}

        query\_image\_path\_normalized = os.path.*normpath*(os.path.*abspath*(*query\_image\_path*))

        for image\_path, histogram in *self*.histograms.*items*():

*# 规范化图像路径*

            image\_path\_normalized = os.path.*normpath*(os.path.*abspath*(image\_path))

*# 跳过原图本身*

            if query\_image\_path\_normalized == image\_path\_normalized:

                continue

*# SIFT相似度*

            cosine\_sim = *cosine\_similarity*([query\_histogram], [histogram])[0][0]

            euclidean\_dist = 1- np.linalg.*norm*(query\_histogram - histogram) */* np.sqrt(*self*.vocab\_size)

            chi2\_dist = 1 - *sum*([(a - b) \*\* 2 / (a + b + 1e-10) for a, b in *zip*(query\_histogram, histogram)]) / 2

*# HOG相似度*

            hog\_vector = *self*.*hog\_features*[*image\_path*].reshape(1, -1)

            hog\_sim = *cosine\_similarity*(query\_hog\_vector, hog\_vector)[0][0]

*# 综合相似度 (可根据效果调整权重)*

            similarity = 0.6\*cosine\_sim + 0.4\*hog\_sim

*# similarity = 0.7\*cosine\_sim + 0.2\*euclidean\_dist + 0.1\*chi2\_dist*

*similarities*[*image\_path*] = similarity

*# 按相似度降序排序*

        sorted\_results = *sorted*(similarities.*items*(), *key*=lambda *x*: *x*[*1*], *reverse*=True)

        return *sorted\_results*[:*top\_n*]

1. **实验结果截图**

本实验使用caltech-101数据集，该数据集主要用于目标识别、图像分类和图像检索任务。其中每个类别有数十张图片

**1.检索brain下的与image\_0018.jpg相似的数据集中的图片：**



2.检索windsor\_chair下的椅子图片：



可以看出该模型在形状上的相似度检测较为准确。

**c) 实验小结**

本实验只在实现一个基于SIFT(Scale-Invariant Feature Transform)特征提取和词袋模型(Bag of Visual Words, BoVW)的图像检索系统。SIFT特征对于图像变换具有鲁棒性，我在训练完后，及时保存了模型的数据字典，避免了每次启动任务都要训练模型的繁琐步骤，提升了计算效率。同时，我添加了HOG特征提取，来综合考量不同维度的特征，计算更加准确的相似度。但与此同时，图像的复杂背景会引入噪声特征，减低检索精度。后续可以通过结合CNN深度学习特征，提升语义理解能力。