# 计算机视觉期末作业文档

目录

[计算机视觉期末作业文档 1](#_Toc29111)

[1、 Harris角点检测算子 2](#_Toc30220)

[1.1图像预处理： 2](#_Toc4268)

[1.2计算图像梯度： 2](#_Toc24228)

[1.3计算角点响应函数的输入： 2](#_Toc7459)

[1.4计算角点响应值： 2](#_Toc8860)

[1.5阈值处理： 2](#_Toc24770)

[1.6非极大值抑制： 2](#_Toc18001)

[1.7处理边界： 2](#_Toc21135)

[2、 关键点的描述及其匹配 3](#_Toc24651)

[2.1 生成描述向量 3](#_Toc9251)

[2.1.1图像预处理： 3](#_Toc17443)

[2.1.2提取角点坐标： 3](#_Toc18808)

[2.1.3初始化描述子矩阵： 3](#_Toc20908)

[2.1.4计算描述子： 3](#_Toc22501)

[2.2 匹配描述子 3](#_Toc29286)

[2.2.1初始化： 3](#_Toc15952)

[2.2.2计算欧氏距离矩阵： 4](#_Toc24448)

[2.2.3判断角点是否匹配： 4](#_Toc10139)

[3. 转换矩阵的估计 4](#_Toc23795)

[3.1构造系统矩阵： 4](#_Toc8667)

[3.2求解线性系统： 4](#_Toc14981)

[4. RANSAC 4](#_Toc14295)

[4.1初始化： 4](#_Toc12709)

[4.2RANSAC迭代： 4](#_Toc22490)

[4.3更新最佳内点集： 5](#_Toc12174)

[4.4输出： 5](#_Toc31213)

[5. 梯度方向直方图 5](#_Toc21773)

[5.1计算梯度： 5](#_Toc18663)

[5.2角度分类： 5](#_Toc472)

[5.3.分块处理： 5](#_Toc27897)

[5.4输出： 5](#_Toc12135)

[6. 更佳的图片融合策略 5](#_Toc19360)

[6.1确定融合区域： 6](#_Toc16284)

[6.2线性融合： 6](#_Toc28707)

[6.3输出结果 6](#_Toc17868)

[7. 运行结果 6](#_Toc10503)

## Harris角点检测算子

### 1.1图像预处理：

将输入的RGB图像转换为灰度图像。初始化一个与输入图像大小相同的矩阵E，用于存储每个像素的角点响应值。

### 1.2计算图像梯度：

使用Sobel算子计算图像在x方向和y方向的梯度值Gx和Gy。

### 1.3计算角点响应函数的输入：

计算图像梯度乘积Gx.\*Gx、Gx.\*Gy和Gy.\*Gy，这些乘积是构建角点响应函数矩阵M的元素。使用高斯滤波器对梯度乘积进行平滑处理，以减少噪声的影响。

### 1.4计算角点响应值：

对于图像中的每个像素，构建2x2的矩阵M，其中M(1,1)=A(i,j)，M(1,2)=M(2,1)=B(i,j)，M(2,2)=C(i,j)。根据角点响应函数R = det(M) - k \* (trace(M))^2计算每个像素的角点响应值，其中det(M)是矩阵M的行列式，trace(M)是矩阵M的迹（对角线元素之和）。参数k通常取0.04到0.06之间的值，用于调节响应函数的灵敏度。

### 1.5阈值处理：

找到角点响应值中的最大值Emax，并设置阈值t为0.01 \* Emax。将响应值小于阈值的像素置为0，认为它们不是角点。

### 1.6非极大值抑制：

对于每个非零响应值，检查其是否为其8邻域（包括自身）中的最大值。如果不是，则将其置为0。

### 1.7处理边界：

由于边缘效应，图像边界上的像素通常不包含有效的角点信息。因此，将图像边界上一定范围内的像素的角点响应值置为0。

## 关键点的描述及其匹配

### 2.1 生成描述向量

#### 2.1.1图像预处理：

首先将其转换为灰度图像，以便进行后续处理。这是因为在灰度图像上更容易计算特征。

#### 2.1.2提取角点坐标：

使用find函数在角点图corners中查找值为1的像素位置，这些位置对应于检测到的角点。将这些角点的坐标存储在keypoints矩阵中，每行包含一个角点的(x, y)坐标。

#### 2.1.3初始化描述子矩阵：

根据patch\_size计算每个patch的半径ps（取整）。初始化descriptors矩阵，用于存储每个角点的描述子。

#### 2.1.4计算描述子：

对于每个角点，根据其坐标在灰度图像上提取一个(patch\_size\*2) x (patch\_size\*2)大小的patch.将这个patch分成4个(patch\_size x patch\_size)的小块，分别提取并组合成一个大的(2\*patch\_size) x (2\*patch\_size)的patch。将计算得到的描述子存储在descriptors矩阵的相应行中。

### 2.2 匹配描述子

#### 2.2.1初始化：

descriptors1 和 descriptors2 分别是两个图像中角点的描述子矩阵，其中 m 和 n 分别是两个图像中角点的数量，d 是描述子的维度。k 是比例阈值，用于判断两个描述子是否足够接近以被认为是匹配的。matched\_points 初始化为一个 m x 2 的零矩阵，用于存储匹配上的角点的索引（第一列是 descriptors1 中的索引，第二列是 descriptors2 中的索引）。dist 初始化为一个 m x n 的零矩阵，用于存储两个描述子矩阵之间所有可能的欧氏距离。

#### 2.2.2计算欧氏距离矩阵：

使用两层循环遍历 descriptors1 和 descriptors2 中的所有描述子对。对于每一对描述子，计算它们之间的欧氏距离，并将结果存储在 dist 矩阵的相应位置。

#### 2.2.3判断角点是否匹配：

对于 descriptors1 中的每个描述子，在 dist 矩阵的相应行中找到最小的两个距离（即与 descriptors2 中两个最接近的描述子的距离）。计算最小距离与次小距离的比例 q。如果 q 小于或等于比例阈值 k，则认为 descriptors1 中的当前描述子与 descriptors2 中对应最小距离的描述子是匹配的。将匹配的角点索引存储在 matched\_points 矩阵中。

## 转换矩阵的估计

### 3.1构造系统矩阵：

对于每一对匹配点 (xi​,yi​) 和 (ui​,vi​)，构造一个 2×6 的矩阵 tempA（或 6×2 的矩阵，取决于实现），并将其添加到 A 中。同时，构造一个 2×1 的向量 tempb（或 2×1 的向量，如果直接使用 B），并将其添加到 b（或 B）中。

### 3.2求解线性系统：

使用最小二乘法求解线性系统 Ah=b计算 (A′A)−1A′b。输出拟合的仿射变换矩阵 H。

## RANSAC

### 4.1初始化：

初始化ransac\_matched\_points为零矩阵，用于存储最终的匹配点对。提取matched\_points中的匹配点索引，并转换为对应的坐标。

### 4.2RANSAC迭代：

进行指定次数的迭代（iterations）。在每次迭代中，随机选择num\_inliers个点对作为内点候选。使用这些候选内点拟合一个仿射变换矩阵H。应用这个矩阵H将图像1中的所有匹配点变换到图像2的空间中。计算变换后的点与对应原始点之间的误差。根据误差阈值thres判断哪些点是内点（即误差小于阈值的点）。

### 4.3更新最佳内点集：

如果当前迭代找到的内点数量大于之前迭代找到的内点数量，则更新最佳内点集和对应的仿射变换矩阵。

### 4.4输出：

迭代结束后，输出最佳内点集（即剔除误匹配后的匹配点）和对应的内点数量。移除ransac\_matched\_points中全零的行（如果有的话），因为这些行表示没有匹配上的点。

## 梯度方向直方图

### 5.1计算梯度：

使用Sobel算子计算图像块在x方向和y方向的梯度Gx和Gy。计算梯度幅值G和梯度方向theta。梯度幅值通过Gx和Gy的平方和的平方根得到，梯度方向通过atan2d(Gy, Gx)计算得到

### 5.2角度分类：

定义直方图的bin数量n\_bin（这里将360度分为8个bin，即每45度一个bin）。使用sorter函数根据梯度方向theta将每个像素分类到对应的bin中。由于梯度方向是连续的，一个像素的梯度可能跨越两个bin的边界，因此每个像素会被分配到两个相邻的bin中，并计算其在这两个bin中的权重。

### 5.3.分块处理：

将图像块patch划分为多个小的“cell”，每个cell的大小是2\*2对每个cell计算梯度直方图。由于一个像素的梯度可能跨越两个bin，因此每个cell的直方图是通过累加每个像素在两个相邻bin中的贡献来得到的。

### 5.4输出：

将所有cell的直方图连接成一个长向量hog\_descriptors，作为该图像块的HOG特征描述子。

## 更佳的图片融合策略

### 6.1确定融合区域：

根据 img1 和 img2 有无像素值确定融合区域 temp\_mask；

### 6.2线性融合：

创建一个线性融合的掩膜（linear\_blend\_mask），该掩膜在重合区域从左到右逐渐从1过渡确定融合区域的左边界和右边界； 创建比重相同融合 mask1 和线性融合 mask2； 根据 mask 计算每张图的重合区域像素值；

### 6.3输出结果

将线性融合后的img1和img2（尽管它们的掩膜设置是错误的）相加，得到线性融合的结果（linear\_blended\_img）。将比重相同融合后的两幅图像相加，得到比重相同融合的结果

## 运行结果





