Canny 边缘检测与霍夫圆检测算法设计文档

1. 算法整体流程

1.1 Canny 边缘检测流程

1. **高斯滤波**：使用高斯核对输入图像进行平滑，以降低图像中噪声对检测结果的干扰。

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

   其中：
   - $G(x,y)$ 是高斯核在位置 (x,y) 处的值。
   - σ 是高斯分布的标准差，决定了核的宽度和平滑程度。
   - x 和 y 是相对于核中心的位置坐标。
2. **梯度计算**：使用 Sobel 算子计算图像的梯度幅度和方向。
   - Sobel算子分别在x和y方向计算梯度。
     - Sobel 算子在 x 方向的核为：

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

     - Sobel 算子在 y 方向的核为：

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

   - 计算图像的梯度幅值：$G = \sqrt{G_X^2 + G_Y^2}$
   - 计算梯度方向: $\theta = \arctan\left(\frac{G_Y}{G_X}\right)$
3. **非极大值抑制**：沿着梯度方向抑制非极大值，保留边缘点。
   - 对梯度方向的8邻域的梯度幅值进行比较，保留最大值。
4. **双阈值检测**：通过高低阈值筛选出强边缘和弱边缘。
5. **边缘追踪**：通过深度优先搜索（DFS）或广度优先搜索（BFS）连接弱边缘和强边缘，形成完整的边缘。
   - 当弱边缘连接强边缘时，匠弱边缘置为强边缘，否则丢弃。

1.2 霍夫圆检测流程

1. **边缘检测**：使用 Canny 算法获取边缘图像。
2. **霍夫投票**：在霍夫空间中对边缘点进行投票，检测可能的圆心和半径。
   - 建立霍夫空间：对圆检测来说，是将空间(x,y)中的点映射到空间(a,b,r)，分别对应圆公式中的：

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

     其中a,b,r未知。
     映射之后霍夫空间中表示为:

$$(a - x_i)^2 + (b - y_i)^2 = r^2$$

     此时x_i, y_i为圆心，a,b,r分别是圆心的坐标和半径。在霍夫空间中，对同一对x_i, y_i，呈现的图形为圆锥，圆锥上的每一点对应一对可能的a,b,r。
   - 投票：对于x,y空间的圆，圆上的每一点都可以在霍夫空间中映射出一个圆锥，这些圆锥都会满足同一个a,b,r，即相交点。由于a,b,r都未知，我们可以对所有圆锥上的点进行投票，最后根据投票阈值筛选出最可能的结果。也就是对任意的x_i, y_i，$houghspace[a,b,r] + 1$。
   - 霍夫梯度：由于三维空间中圆锥点的数量过多，计算量会很大，所以我们只计算梯度方向的可能点，因为法线一定会穿过圆点。
   - 离散化：由于图像空间是离散的，所以霍夫空间也是离散的，在实际计算过程中，我们可以设置步长$step$来降低计算量。对应的更新公式为：

$$x_{i+1} = x_i \pm step \qquad y_{i+1} = y_i \pm step \times \tan\theta$$
$$r_{i+1} = r_i \pm \sqrt{\Delta x^2 + \Delta y^2}$$

3. **非极大值抑制**：去除重叠的圆，保留最终的检测结果。

2. 函数功能说明

2.1 canny.py

2.1.1 `__init__` 方法

- **功能**：初始化Canny算子。
- **输入**：
  - `gaussian_kernel_size`： 高斯核大小（ `int` ）。
  - `sigma`： 高斯分布标准差（ `float` ）。
  - `low_threshold`： 双阈值中的低阈值（ `int` ）。
  - `high_threshold`： 双阈值中的高阈值（ `int` ）。

2.1.2 `__call__` 方法

- **功能**：执行 Canny 边缘检测的完整流程。
- **输入**：
  - `image`： 输入的灰度图像（ `np.ndarray` ）。
- **输出**：
  - `output_img`： 检测到的边缘图像（ `np.ndarray` ）。
  - `gradient_angle`： 梯度方向矩阵（ `np.ndarray` ）。

2.1.3 `padding` 方法

- **功能**：对图像进行零填充，以适应卷积操作。
- **输入**：
  - `image`： 输入的图像（ `np.ndarray` ）。
- **输出**：
  - `output_img`： 填充后的图像（ `np.ndarray` ）。

2.1.4 `build_gaussian_kernel` 方法

- **功能**：构建高斯核。
- **输出**：
  - `gaussian_kernel`： 高斯核（ `np.ndarray` ）。

2.1.5 `gaussian_blur` 方法

- **功能**：使用numpy循环嵌套卷积对图像进行高斯模糊。
- **输入**：
  - `padded_img`： 填充后的图像（ `np.ndarray` ）。
- **输出**：
  - `output_img`： 高斯模糊后的图像（ `np.ndarray` ）。

2.1.6 `gaussian_blur_sripy` 方法

- **功能**：使用 `scipy.signal.convolve2d` 对图像进行高斯模糊。
- **输入**：
  - `padded_img`： 填充后的图像（ `np.ndarray` ）。
- **输出**：
  - `output_img`： 高斯模糊后的图像（ `np.ndarray` ）。

2.1.7 `sobel` 方法

- **功能**：使用numpy循环嵌套卷积应用 Sobel 算子。
- **输入**：
  - `is_horizontal`： 是否为水平方向（ `bool` ）。
  - `image`： 输入图像（ `np.ndarray` ）。
- **输出**：

- `output_img`： Sobel 操作后的图像（ `np.ndarray` ）。

2.1.8 `sobel_scipy` 方法

- **功能**：使用 `scipy.signal.convolve2d` 应用 Sobel 算子。
- **输入**：
  - `is_horizontal`： 是否为水平方向（ `bool` ）。
  - `image`： 输入图像（ `np.ndarray` ）。
- **输出**：
  - `output_img`： Sobel 操作后的图像（ `np.ndarray` ）。

2.1.9 `non_maximum_suppression` 方法

- **功能**：执行非极大值抑制。
- **输入**：
  - `gradient_magnitude`： 梯度幅度矩阵（ `np.ndarray` ）。
  - `gradient_angle`： 梯度方向矩阵（ `np.ndarray` ）。
- **输出**：
  - `gradient_magnitude_nms`： 非极大值抑制后的梯度幅度矩阵（ `np.ndarray` ）。

2.1.10 `double_threshold_filter` 方法

- **功能**：执行双阈值检测。
- **输入**：
  - `image`： 输入图像（ `np.ndarray` ）。
- **输出**：
  - `output_img`： 双阈值检测后的图像（ `np.ndarray` ）。

2.1.11 `edge_tracking` 方法

- **功能**：通过 BFS 接弱边缘和强边缘，形成完整的边缘。。
- **输入**：
  - `image`： 输入图像（ `np.ndarray` ）。
- **输出**：
  - `output_img`： 最终的边缘图像（ `np.ndarray` ）。

2.2 hough_circle.py

2.2.1 `__init__` 方法

- **功能**：初始化霍夫圆检测算法。
- **输入**：
  - `step_length`： 步长（ `int` ）。
  - `vote_threshold`： 投票阈值（ `int` ）。
  - `distance_threshold`： 距离阈值，与圆心间距比较判断是否重叠（ `int` ）。

2.2.2 `__call__` 方法

- **功能**：执行霍夫圆检测的完整流程。
- **输入**：
  - `image`： 输入的边缘图像（ `np.ndarray` ）。
  - `angle_matrix`： 梯度方向矩阵（ `np.ndarray` ）。
  - `step_length`： 步长（ `int` ）。
  - `vote_threshold`： 投票阈值（ `int` ）。
  - `distance_threshold`： 梯度方向矩阵（ `int` ）。
- **输出**：
  - `final_circles`： 检测到的圆的列表（ `np.ndarray` ），形状为 `[ [y, x, r], ... ]`。

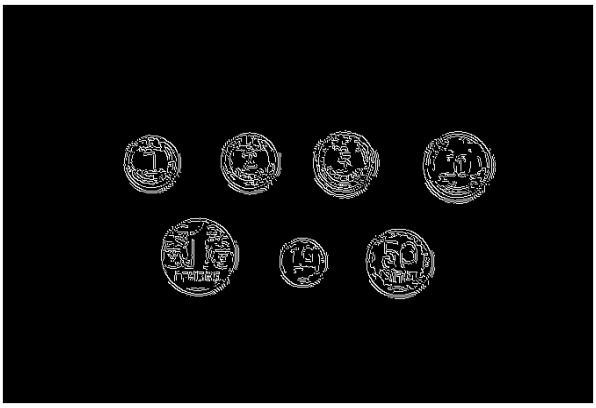
2.2.3 `vote` 方法

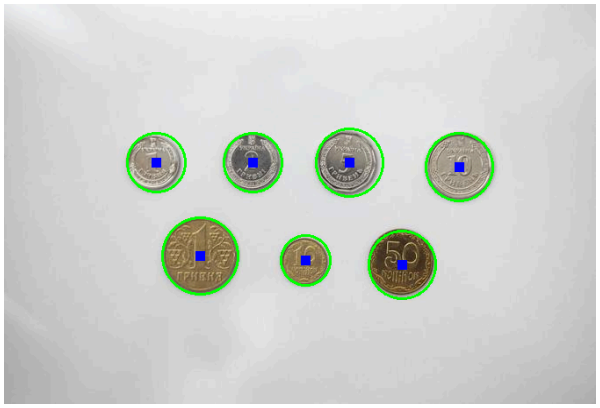
- **功能**：在霍夫空间中对边缘点进行投票。
- **输入**：
  - `image`： 输入的边缘图像（ `np.ndarray` ）。
  - `angle_matrix`： 梯度方向矩阵（ `np.ndarray` ）。
- **输出**：
  - `circles`： 投票结果列表（ `list` ），形状为 `[ [y, x, r], ... ]`。

2.2.4 `non_maximum_suppression` 方法

- **功能**：去除重叠的圆。
- **输入**：
  - `circles`： 检测到的圆的列表（ `list` ）。
- **输出**：
  - `final_circles`： 最终检测到的圆的列表（ `np.ndarray` ）。

4. 最终拟合结果图





5. 参数对最终定位结果的影响分析

5.1 Canny 边缘检测

- **`gaussian_kernel_size`**：
  - 高斯核 size 增大时，图像的高频信息（如边缘）会被更多地过滤掉。这可能导致边缘检测时，边缘的位置不准确，进而影响霍夫圆检测中圆心和半径的计算。导致检测的圆偏移。
- **`sigma`**：
  - 较大的标准差会增加平滑程度，但可能导致边缘丢失。
  - 较小的标准差保留更多细节，但可能保留过多噪声。
- **`low_threshold` 和 `high_threshold`**：
  - 较低的阈值会检测到更多的边缘，但可能包含噪声。
  - 较高的阈值会检测到更清晰的边缘，但可能丢失一些细节。

5.2 霍夫圆检测

- **`step_length`**：
  - 较小的步长会提高检测精度，但增加计算量。
  - 较大的步长会降低计算量，但可能错过一些圆。
- **`vote_threshold`**：
  - 较低的投票阈值会检测到更多的圆，但可能包含误检。
  - 较高的投票阈值会减少误检，但可能错过一些真正的圆。
- **`distance_threshold`**：
  - 较小的距离阈值会减少重叠圆的检测，但可能导致漏检。
  - 较大的距离阈值会增加检测范围，但可能检测到重叠的圆。