Multi Sensor fusion for In-process Control

Acoustic Signal Map with ROI

Documents and Function Guide for Acoustic Signal Map with ROI

Ming Wu

KU Leuven

2024-11-06

1 Introduction

The `signal_within_radius` function identifies points within a specified radius (Region of Interest, ROI) around a central point and compresses information about these points for further use. This information can then be mapped to a data acquisition (DAQ) map, allowing integration of geometrical information of a printing object.

2 Function: signal_within_radius

2.1 Algorithm

Algorithm 1 Signal Within Radius

```
function SIGNAL_WITHIN_RADIUS(xy_map, x_center, y_center, radius, daq_map_lengths, lmq_map_lengths)
    neighbors ← find_neighbors_within_radius(xy_map, x_center, y_center, radius)
    compressed_info ← compress_neighbor_info(neighbors)
    daq_mapped_info ← map_to_daq_map(compressed_info, daq_map_lengths, lmq_map_lengths)
    daq_mapped_info.sort(key=lambda n: n[0])
    return daq_mapped_info
end function
```

2.2 Mathematical Explanation

Given a center point (x_{center}, y_{center}) and a radius r, the function computes the compressed information for the points within the radius.

The distance from a point (x, y) to the center can be mathematically expressed as:

$$d = \sqrt{(x - x_{center})^2 + (y - y_{center})^2}$$

The condition for points to be within the radius is:

$$d \leq r$$

The angles are calculated with respect to the center using:

$$\theta = \tan^{-1}\left(\frac{y - y_{center}}{x - x_{center}}\right)$$

3 Function: calculate_relative_point

3.1 Algorithm

Algorithm 2 Calculate Relative Point

```
function CALCULATE_RELATIVE_POINT(x0, y0, distance, angle)
    x ← x0 + distance × cos(angle)    y← y0 + distance × sin(angle)
    return (x, y)
end function
```

3.2 Mathematical Explanation

The function computes the position of a point based on a reference point (x_0, y_0), a distance d, and an angle θ:

$$x = x_0 + d \cdot \cos(\theta)$$
$$y = y_0 + d \cdot \sin(\theta)$$

where d is the Euclidean distance from the reference point to the new point.

4 Function: calculate_distance

4.1 Algorithm

Algorithm 3 Calculate Distance

```
function CALCULATE_DISTANCE(p1, p2)
    return sqrt((p2[0] - p1[0])^2 + (p2[1] - p1[1])^2)end function
```

4.2 Mathematical Explanation

The distance d between two points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ is calculated using the Euclidean formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

5 Function: find_longest_pair

5.1 Algorithm

Algorithm 4 Find Longest Pair

```
function FIND_LONGEST_PAIR(point1_list, point2_list)
    distances ← []
    for i = 1 to length(point1_list) do
        d ← calculate_distance(point1_list[i], point2_list[i])
        distances.append(d)
    end for
    longest_pair_index ← index_of_max(distances)
    return longest_pair_index
end function
```

5.2 Mathematical Explanation

The function identifies the pair of points with the maximum distance:

$$\text{max\_distance} = \max_i(d_i) \quad \text{where} \quad d_i = \sqrt{(x_i^{(1)} - x_i^{(2)})^2 + (y_i^{(1)} - y_i^{(2)})^2}$$

The index of the longest pair is returned.

6 Function: rotate_points

6.1 Algorithm

Algorithm 5 Rotate Points

function ROTATE_POINTS(points, angle)
    rotation_matrix $\leftarrow \begin{bmatrix} \cos(angle) & -\sin(angle) \\ \sin(angle) & \cos(angle) \end{bmatrix}$
    rotated_points $\leftarrow$ np.dot(points, rotation_matrix^T) **return***rotated_points*

6.2 Mathematical Explanation

To rotate a point (x, y) about the origin by an angle θ:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

The resulting coordinates (x', y') are the new coordinates after rotation.

7 Function: geometrical_embedded_offset

7.1 Algorithm

Algorithm 6 Geometrical Embedded Offset

function GEOMETRICAL_EMBEDDED_OFFSET(daq_mapped_info, x_center, y_center)
    longest_pair_index $\leftarrow$ find_longest_pair(s0s, s1s)
    longest_pair $\leftarrow$ point_pairs[longest_pair_index]
    angle $\leftarrow$ atan2($y^{(1)}_{longest\_pair} - y^{(0)}_{longest\_pair}, x^{(1)}_{longest\_pair} - x^{(0)}_{longest\_pair}$) *rotated_point_pairs*$\leftarrow$ rotate_points(point_pairs, -angle)
    ... [further computations]
    return first_line_direction, start_offset
end function

7.2 Mathematical Explanation

The function computes offsets to geometrically align multiple points relative to the longest vector:

1. Determine the angle of the longest vector:

$$\theta = \tan^{-1}\left(\frac{y_2 - y_1}{x_2 - x_1}\right)$$

2. Rotate all points counter-clockwise by $-\theta$ using the rotation matrix.
3. Compute offsets:

$$\text{offset} = x_{rotated} - \min(x_{rotated})$$

This embeds geometric information through relative offsets.

8 Function: place_signals_on_grid

8.1 Description

Interpolate each signal onto a uniform grid and arrange them in a 2D array.

8.2 Algorithm

Algorithm 7 place_signals_on_grid

function PLACE_SIGNALS_ON_GRID(signals, t_data, max_length, resolution, fill)
    Define a uniform time grid
    for each signal **do**
        Interpolate and append to result
    end for
    Stack interpolated signals into a 2D array
    return signal_matrix
end function

8.3 Mathematical Explanation

The uniform time grid can be defined as:

$$t_j = j \times resolution,\ j = 0, 1, \ldots, \left\lceil \frac{max\_length}{resolution} \right\rceil$$

The interpolation is performed for each signal using linear interpolation.

9 Function: from_xy_get_signal_matrix

9.1 Description

Combines the functionality of previous functions to produce a matrix of signals.

9.2 Algorithm

Algorithm 8 from_xy_get_signal_matrix

function FROM_XY_GET_SIGNAL_MATRIX(center_x, center_y, xy_map, daq_map_lengths, lmq_map_lengths, ae_collection, sampling_rate_daq, max_length, resolution, fill)
    daq_mapped_info ← SIGNAL_WITHIN_RADIUS(...)
    if daq_mapped_info is empty **then**
        return zero_matrix
    end if
    first_line_direction, start_offset ← GEOMETRICAL_EMBEDDED_OFFSET(...)
    for each info in daq_mapped_info **do**
        Process and compile signals
    end for
    return signal_matrix
end function

9.3 Mathematical Explanation

Final output consists of a matrix formed from signals based on offsets. Each entry can be defined as:

$$signal\_matrix_{i,j} = |ae_{i,j}|$$

where $ae_{i,j}$ is the j^{th} signal of the i^{th} line.