Distance Peak Detector

User Guide

A111 – Distance Peak Detector

User Guide

Author: Acconeer

Version 2.0: 2018-07-04

Acconeer AB

Table of Contents

1 Introduction

The distance detector is implemented on top of the envelope service. From SW version 1.35, the detector API is changed so that the user do not have to access the envelope API explicitly to fetch data and pass it to the envelope API. Instead the Distance Peak Detector will call the envelope service API whenever it needs new data to process.

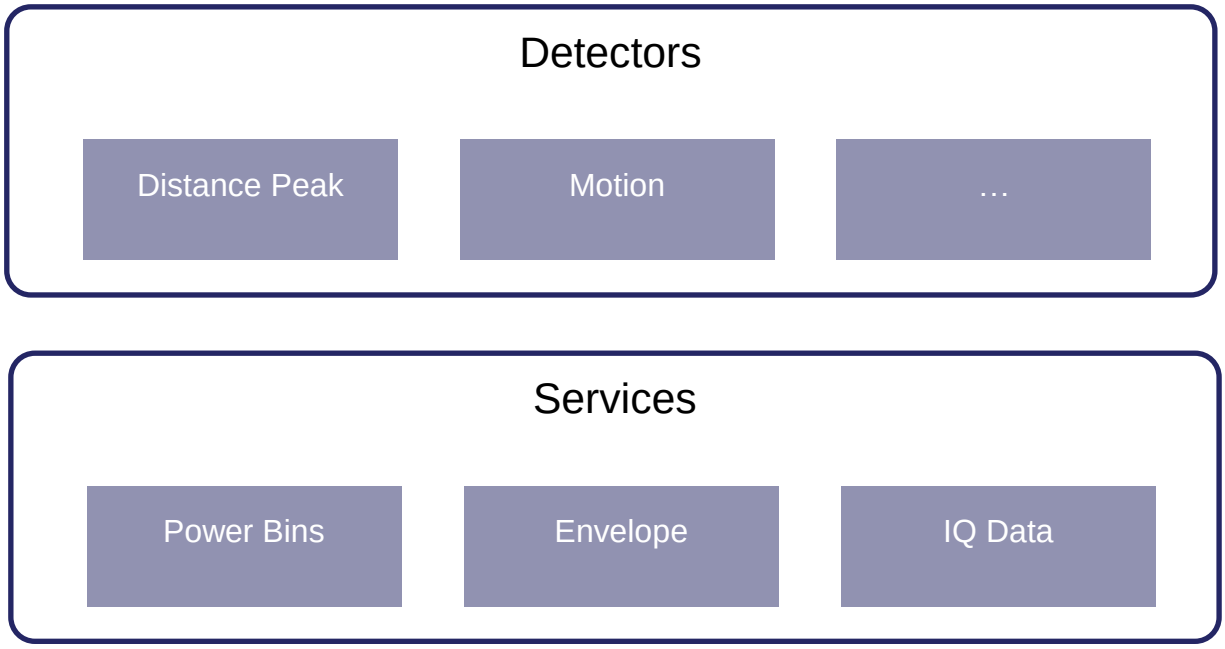

Figure 1- Acconeer SW Interfaces

2 Configuring the Distance Peak Detector

To use the Distance Peak Detector, first a configuration must be created. To create a configuration, call the **acc_detector_distance_peak_configuration_create** function which will create a configuration and return it.

```
if (!acc_rss_activate()) {
            /* Handle error */
}

acc_detector_distance_peak_configuration_t distance_configuration;

distance_configuration = acc_detector_distance_peak_configuration_create();

acc_detector_distance_peak_handle_t handle =
acc_detector_distance_peak_create(distance_configuration);
```

A newly created configuration is populated with default parameters and can be used directly to create the detector by calling the **acc_detector_distance_peak_create** function. A more common scenario is to first modify some of the configuration parameters to better fit the application.

2.1 Setting Threshold Mode

2.1.1 Fixed Threshold Mode

In fixed threshold mode, you can specify the minimum amplitude level to detect. Any object reflections with an amplitude below the minimum level, will be ignored by the detector. To configure the Distance Peak Detector to operate in fixed threshold mode, call the `acc_detector_distance_peak_set_threshold_mode_fixed` function.

```
acc_detector_distance_peak_status_t detector_status;
acc_detector_distance_peak_configuration_t distance_configuration;

distance_configuration = acc_detector_distance_peak_configuration_create();

detector_status = acc_distance_set_detector_threshold_mode_fixed(distance_configuration,
                                                                 FIXED_THRESHOLD_VALUE);
```

2.1.2 Stationary Clutter Threshold Mode

In “stationary clutter estimated threshold” mode, first the detector records background reflections from stationary objects in the environment surrounding the sensor. A threshold varying with distance is then calculated, so that the amplitude of the reflections from the stationary objects will be located below the threshold level at the distances where the objects are located. At distances with no stationary clutter, the threshold level will be lower.

To set up a detector in this mode call the **acc_detector_distance_peak_threshold_estimation_update** function.

You are recommended to use at least 50 updates with background reflections containing stationary clutter before using the Distance Peak Detector.

A new threshold estimation should be performed if significant changes were made in the sensor's surrounding environment. To reset the Distance Peak Detector to empty state, please call the **acc_detector_distance_peak_threshold_estimation_reset** function. Then update the Distance Peak Detector for the new environment using the **acc_detector_distance_peak_threshold_estimation_update** function.

```
acc_detector_distance_peak_status_t detector_status;

acc_detector_distance_peak_get_metadata(handle, &metadata);
float start_m = metadata.actual_start_m;
float end_m = metadata.actual_start_m + metadata.actual_length_m;


detector_status =
acc_detector_distance_peak_threshold_estimation_update(distance_configuration,
100, metadata.actual_start_m,metadata.actual_start_m + metadata.actual_length_m);
```

It is possible to control the sensitivity and false detection rate of the Distance Peak Detector in estimated threshold mode. With high sensitivity, the detector is more likely to make false detections, e.g. interpret noise as an object. At the same time, the number of missed detections is low. With low sensitivity, the number of missed detections is likely to increase, whereas false detections are likely to decrease.

The sensitivity of the detector is set when calling the **acc_detector_distance_peak_set_sensitivity** function. This function takes a sensitivity parameter in the range between 0 and 1, where the 0 represents the lowest sensitivity and 1 the highest. The function is optional but must be called before the first before activating the detector.

2.2 Setting Sweep Parameters

The sweep configuration parameters determine the sensor source and how the sweep data will be generated in the sensor. The sweep configuration parameters are common to all services and are also possible to set in the Distance Peak Detector. Like other configuration parameters, the sweep parameters have reasonable default values, but in most applications, it is necessary to modify at least some of them. To do this we must first obtain a sweep configuration handle.

```
acc_sweep_configuration_t sweep_configuration;

sweep_configuration = acc_service_get_sweep_configuration(distance_configuration);

if (sweep_configuration == NULL) {
    /* Handle error */
}
```

Using the sweep configuration handle, we can use access functions to set individual configuration parameters such as the sweep range and frequency.


```
// Set sweep start and length
acc_sweep_configuration_requested_range_set(sweep_configuration, .20, 0.4);

// Set sweep frequency
acc_sweep_configuration_repetition_mode_streaming_set(sweep_configuration, 100);
```

See the “Envelope Service User Guide” for a more complete explanation of the sweep parameters.

2.3 Adjusting the Running Average for Better Accuracy

The range and accuracy of distance measurements can be improved when running the detector using an average of multiple sweeps. This procedure may be controlled by calling the function **acc_detector_distance_peak_running_average_factor_set**. By setting the “factor” parameter to a value between 0-1 where 0 means that averaging is disabled and 1 means that the first detection is always returned.

The current default value for this setting is 0.7. When measuring objects in motion this value may be decreased. To improve SNR for static objects the running average factor could be increased to a value closer to 1.

```
acc_detector_distance_peak_configuration_t distance_configuration;
float factor = 0.9f;

distance_configuration = acc_detector_distance_peak_configuration_create();
acc_detector_distance_peak_running_average_factor_set(distance_configuration, factor);
```

3 Measure Distances

3.1 Creating and Activating the Distance Peak Detector

To activate the detector call the **acc_detector_distance_peak_activate** function. Now the detector is producing detector data which might be retrieved by calling the **acc_detector_distance_peak_get_next** function.

```
acc_detector_distance_peak_handle_t handle =
acc_detector_distance_peak_create(distance_configuration);

detector_status = acc_detector_distance_peak_activate(handle);
if (detector_status != ACC_DETECTOR_DISTANCE_PEAK_STATUS_SUCCESS) {
             /* Handle error */
}
```

3.2 Getting Detection Results

When the detector has been created and activated the detections results may be retrieved by calling the **acc_detector_distance_peak_get_next** function.

```
acc_detector_distance_peak_status_t detector_status;

uint_fast16_t reflection_count = 10;

acc_detector_distance_peak_reflection_t reflections[reflection_count];
acc_detector_distance_peak_configuration_t distance_configuration;
distance_configuration = acc_detector_distance_peak_configuration_create();

acc_detector_distance_peak_handle_t handle;
handle = acc_detector_distance_peak_create(distance_configuration);

detector_status = acc_detector_distance_peak_activate(handle);

detector_status = acc_detector_distance_peak_get_next(handle,
reflections,&reflection_count,
```

To get the actual distances, we must start by allocating memory for an array of type **acc_detector_distance_peak_reflection_t**, for storing distance estimations. In the example above, this array is allocated on the stack. Then we can call **acc_detector_distance_peak_get_next** to fill the array with distances and amplitudes for such reflections, which have been detected as objects by the Distance Peak Detector.

4 Deactivating and Destroying the Distance Peak Detector

To release the memory resources allocated by the Distance Peak Detector, please call the **acc_detector_distance_peak_deactivate** function followed by the **acc_detector_distance_peak_destroy** function and finally by calling the **acc_detector_distance_peak_configuration_destroy** function. Do this when you have reached the point where you do not need to use the detector anymore.

```
detector_status = acc_detector_distance_peak_deactivate(handle);

acc_detector_distance_peak_destroy(&handle);

acc_detector_distance_peak_configuration_destroy(&distance_configuration);
```

5 Additional Tips and Tricks

5.1 Measuring Absolute Distances

There is a small offset error in distances returned by the distance sensor. This may be caused by multiple factors, such as the placement of the sensor in relation to the ground plane, materials covering the sensor and manufacturing process variations.

The sensor specific offset error can be reduced when subtracting the **free_space_absolute_offset**, returned as a result from the call to **acc_sensor_preparation_receive**. To compensate for other sources of offset error, related to the placement of the sensor and surrounding materials, the offset error can be estimated to:

offet_error = a * free_space_absolute_offset + b

The constants **a** and **b** are design specific and depend on electrical environmental factors, such as PCB layout and materials covering the sensor.

5.2 Absolute amplitude

As of release SW v1.1.28, the amplitude values returned by the Distance Peak Detector constitute the difference between the reflection amplitude and the threshold. The **acc_detector_distance_peak_set_absolute_amplitude** function can be called to configure the Distance Peak Detector to legacy behavior and return absolute amplitude values.

```
acc_detector_distance_peak_set_absolute_amplitude(distance_configuration, true);
```


Disclaimer

The information herein is believed to be correct as of the date issued. Acconeer AB ("**Acconeer**") will not be responsible for damages of any nature resulting from the use or reliance upon the information contained herein. Acconeer makes no warranties, expressed or implied, of merchantability or fitness for a particular purpose or course of performance or usage of trade. Therefore, it is the user's responsibility to thoroughly test the product in their particular application to determine its performance, efficacy and safety. Users should obtain the latest relevant information before placing orders.

Unless Acconeer has explicitly designated an individual Acconeer product as meeting the requirement of a particular industry standard, Acconeer is not responsible for any failure to meet such industry standard requirements.

Unless explicitly stated herein this document Acconeer has not performed any regulatory conformity test. It is the user's responsibility to assure that necessary regulatory conditions are met and approvals have been obtained when using the product. Regardless of whether the product has passed any conformity test, this document does not constitute any regulatory approval of the user's product or application using Acconeer's product.

Nothing contained herein is to be considered as permission or a recommendation to infringe any patent or any other intellectual property right. No license, express or implied, to any intellectual property right is granted by Acconeer herein.

Acconeer reserves the right to at any time correct, change, amend, enhance, modify, and improve this document and/or Acconeer products without notice.

This document supersedes and replaces all information supplied prior to the publication hereof.