DSP PLAN FOR INSECT DETECTION

Data Claned

For the CO2 sensor, we used the Hampel Filter for Outlier Detection. With this filter, we used 10 data points as a window and removed extreme values above 99.5% and lower than 0.5% using Percentile Clipping.

Hampel Filter is particularly effective for time series data and only removes points that deviate significantly, which can maintain legitimate rapid changes in the data.

Example code that I used in data cleaning:

```
def hampel_filter(series, window_size=10, n_sigmas=3):
    """
    Apply Hampel filter for outlier detection and removal, followed by
percentile clipping
    Parameters:
        series: time series data
        window_size: number of samples in window (default=10)
        n_sigmas: number of standard deviations for threshold (default=3)
    """
    # Apply Hampel filter
    rolling_median = series.rolling(window=window_size,
center=True).median()
    mad = median_abs_deviation(series, nan_policy='omit')
    threshold = n_sigmas * mad
    difference = np.abs(series - rolling_median)
    hampel_filtered = np.where(difference > threshold, rolling_median,
series)

    # Apply percentile clipping to the Hampel filtered data
    upper = np.percentile(hampel_filtered, 99.5)
    lower = np.percentile(hampel_filtered, 0.5)
    final_series = np.clip(hampel_filtered, lower, upper)

    return final_series
```

DSP for Detection

For the alert system, we set up a threshold to detect whether the CO_2, humidity, and temperature have a peak change. With the real-world experiment, we set up 4 lbs of soybeans with 35 mealworms, causing a light infestation simulation(0.4%～0.5%). We extract the change rate of CO_2, humidity, and temperature in the light infestation simulation and decide to use average and the change of rate as the condition to detect whether the grains have insects.

We can see in CO_2 mean level and the standard deviation of changes below, two different conditions(with worms, without worms) have a huge gap. So we set up the average CO_2 level in each 30-minute window and standard deviation of CO_2 change rates within that window.

$$\Delta CO2(t) = CO2(t) - CO2(t-1)$$

$$std(\Delta CO_2) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\Delta CO_2(i) - \overline{\Delta CO_2})^2}$$

If `mean_CO2 > X` **and** `std_CO2_diff > Y` → likely insect activity（X,Y is the threshold）

```
Mean values and changes:

CO2:
Mean value: 818.97
Average change per second: 0.0003
Standard deviation of changes: 0.1088
Maximum increase: 0.5997
Maximum decrease: -8.3955
Number of increases: 3056
Number of decreases: 2689

TEMPERATURE:
Mean value: 23.39
Average change per second: 0.0000
Standard deviation of changes: 0.0011
Maximum increase: 0.0040
Maximum decrease: -0.0160
Number of increases: 4373
Number of decreases: 4258

HUMIDITY:
Mean value: 52.48
Average change per second: 0.0000
Standard deviation of changes: 0.0022
Maximum increase: 0.0087
Maximum decrease: -0.0213
Number of increases: 5119
Number of decreases: 4983
```

With worms

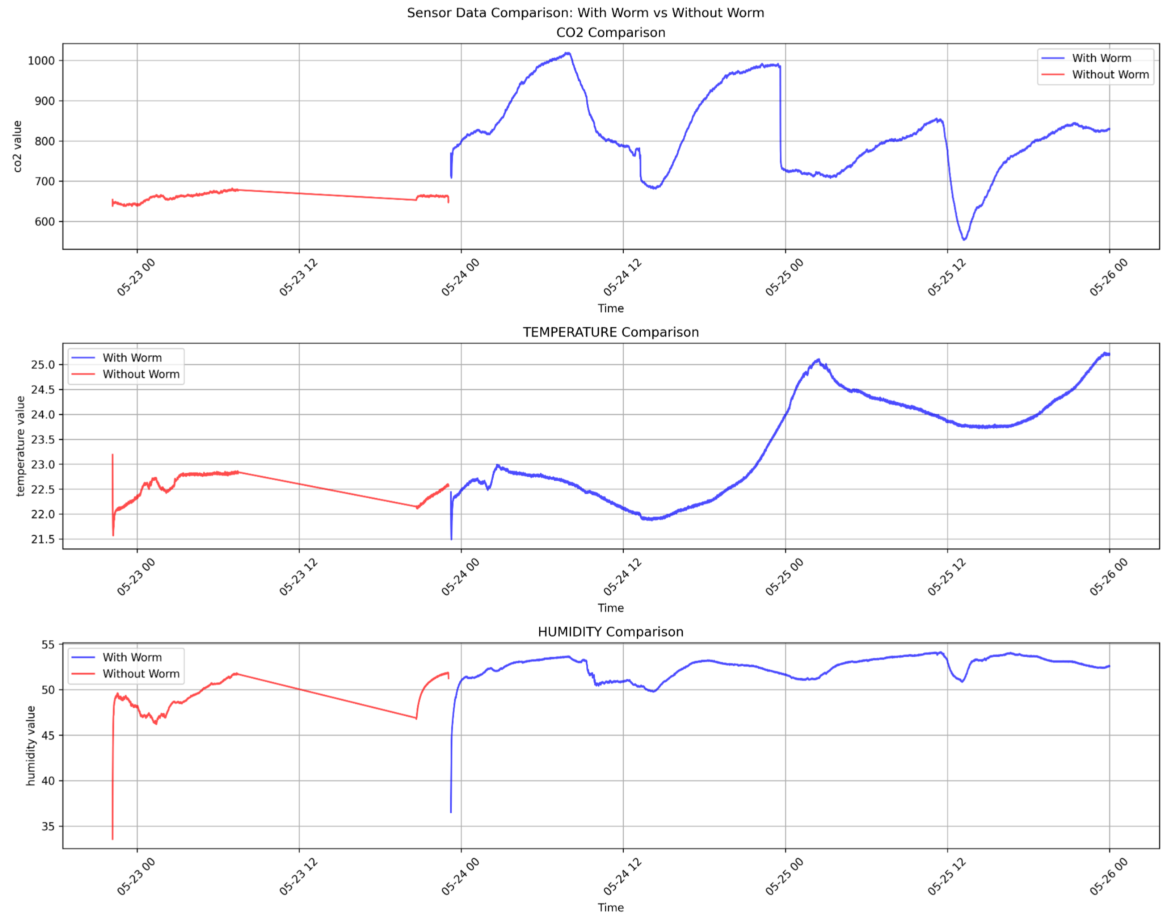
```
Mean values and changes:

CO2:
Mean value: 660.59
Average change per second: 0.0005
Standard deviation of changes: 0.0550
Maximum increase: 0.3332
Maximum decrease: -1.0661
Number of increases: 648
Number of decreases: 614

TEMPERATURE:
Mean value: 22.55
Average change per second: 0.0000
Standard deviation of changes: 0.0012
Maximum increase: 0.0040
Maximum decrease: -0.0227
Number of increases: 1075
Number of decreases: 993

HUMIDITY:
Mean value: 49.34
Average change per second: 0.0002
Standard deviation of changes: 0.0028
Maximum increase: 0.0233
Maximum decrease: -0.0187
Number of increases: 1384
Number of decreases: 1128
```

Without worms



Based on the experiment, for co2 level, all co2 level, standard deviation of changes and peak have huge change between different conditions and can be used to detect whether there is insect in the grains or not.

At first, I used mean_level=660 and std_change=0.05 as the threshold, but the F1 score only is 0.0594 as following shows:

```
Accuracy: 0.2276
Precision: 1.0000
Recall: 0.0306
F1 Score: 0.0594

Confusion Matrix:
True Positives: 3
True Negatives: 25
False Negatives: 95
```

To evaluate our DSP method performance, we use F1 score since our DSP doing classification works, it will label whether the data is classified as having insects or not in the result. So F1 score can show the performance of the method more comprehensively than just using accuracy.

To get those metrics, I use true positives, true negatives, false negatives, false positives to calculate the F1 score and the accuracy.

We have with_worms data and without_worms data, I use those labeled data to test the DSP and get the results.

```
# Evaluate worm data (should detect insects)
    for start in pd.date_range(worm_df['timestamp'].min(),
                               worm_df['timestamp'].max(),
                               freq=window_size):
        end = start + pd.Timedelta(window_size)
        window = worm_df[(worm_df['timestamp'] >= start) &
                         (worm_df['timestamp'] < end)]

        if len(window) > 0:
            detection, _ = detector.analyze_window(window)
            if detection:
                true_positives += 1
            else:
                false_negatives += 1

    # Evaluate without-worm data (should not detect insects)
    for start in pd.date_range(withoutworm_df['timestamp'].min(),
                               withoutworm_df['timestamp'].max(),
                               freq=window_size):
        end = start + pd.Timedelta(window_size)
        window = withoutworm_df[(withoutworm_df['timestamp'] >= start) &
                                (withoutworm_df['timestamp'] < end)]

        if len(window) > 0:
            detection, _ = detector.analyze_window(window)
            if detection:
                false_positives += 1
            else:
                true_negatives += 1

f1_score = 2 * (precision * recall) / (precision + recall)
```

```
accuracy = (true_positives + true_negatives) / total
```

The F1 score for this DSP is too low(only 0.0594), so I try to move to those change:

- Added specific CO_2 change thresholds
- Enhanced CO_2 analysis, added Maximum and minimum change detection, Peak change detection
- Added weighted scoring
    - 50% weight on mean level
    - 30% weight on standard deviation
    - 20% weight on peak changes

Then, in my test, the F1 score rose to 0.8869. And the accuracy rose to 0.7967.

Then, I focus on trying different thresholds and different weights and window size, here's the final results.

```
uments/GitHub/real-world-experiment/src/insect_detection.py

Insect Detection Performance Metrics:
Accuracy: 0.9535
Precision: 1.0000
Recall: 0.9394
F1 Score: 0.9688

Confusion Matrix:
True Positives: 31
False Positives: 0
True Negatives: 10
False Negatives: 2
PS C:\Users\Lily\Documents\GitHub\real-world-experiment>
```

Here's the example code:

```
class InsectDetector:

    def __init__(self):

        # Thresholds based on light infestation experiment (0.4%~0.5%
mealworms)

        self.thresholds = {
```

```
            'mean_level': 680,
            'std_change': 0.05,
            'min_change': 0.3,
            'peak_change': 1.0
        }

    def analyze_window(self, window_data):
        """Analyze a 30-minute window of CO2 data"""
        # Calculate CO2 statistics
        mean_level = window_data['co2'].mean()
        changes = window_data['co2'].diff() /
(window_data['timestamp_ms'].diff() / 1000)
        std_change = changes.std()
        max_change = changes.max()
        min_change = changes.abs().min()

        # Multiple detection criteria for CO2
        level_detected = mean_level > self.thresholds['mean_level']
        std_detected = std_change > self.thresholds['std_change']
        change_detected = (max_change > self.thresholds['peak_change'] or
                         min_change > self.thresholds['min_change'])

        # Calculate detection score (weighted combination)
        score = (level_detected * 0.6 +     # Higher weight for mean level
                std_detected * 0.2 +        # Medium weight for variation
```

```
                change_detected * 0.2)        # Lower weight for peaks

        # Detection threshold lowered for better recall
        detection = score > 0.4
        return detection
```

For working fan detection, we use peak detect. The humidity and temperature did not have a huge change in the experiment. So if the fan cannot work well in the real system, we can see a great change in the data.

For the peak threshold, I calculated the average of change with experiment data as following:

So we use `'temp_max_change': 0.4` and `'humid_max_change': 2.0` as the threshold. See fen_detection.py for the example code.