Offline anomaly detection with causal discovery - Explainable AI

Sebastiano D'Arconso (VR489066) - Tommaso Del Prete (VR488382)

January 2024

1 Introduction

Offline anomaly detection with causal discovery aims to identify unusual or abnormal patterns in data using techniques from causal discovery. In the case of offline detection the analysis is applied to historical data, rather than in real-time. **Anomaly detection** involves identifying patterns in data that devies significantly from the expected behaviour. These anomalies could be indicative of errors, outliers or interesting events in the dataset. **Causal discovery** aims to uncover cause-and-effect relationships within a set of variables. It explores the directionality of relationships between variables, helping understand which variables influence others and in what manner.

2 Datasets

2.1 Swat

The SWAT (Secure Water Treatment) dataset was developed by the Singapore University of Tecnology and Design. This dataset represents a benchmark for anomaly detection, in fact, it represents a unique and challenging example of Cyber-Physical systems (CPS) in a real world example, involving many variables, sensors and conditions. The dataset is organized in two subsets, one **normal** and one **attack**, representing respectively the data collected from the system functioning without anomalies and the data collected during "attacks" (changing environmental conditions in order to create abnormal inputs to the sensors).

2.2 Boat

The boat dataset consists of data collected from autonomous boats for environmental monitoring. The EU-funded project INTCATCH3 focuses on developing cost-effective and user-friendly monitoring strategies for sustainable water quality management using low-cost autonomous surface vehicles (ASVs). These

ASVs navigate autonomously through GPS data. However, concerns about cyber-attacks and hacking are associated with the use of drones and autonomous boats, particularly in the context of water monitoring.

The INTCATCH boats transmit acquired water quality data (e.g., Dissolved Oxygen, pH level, Electrical Conductivity) using standard networks such as Wi-Fi and the 3G mobile network, making the data vulnerable to tampering.

In a specific case study, a training set (DR) of 2,195 normal records was collected during the nominal behavior of the robot, with 659 of them used to estimate thresholds. The test set (TS) consists of 12,280 records distributed among six categories: 660 normal records and the rest from two types of security attacks and faults. The faults include DoS with no payload (19

2.3 Pepper

The pepper dataset consists of data collected from social robots in public environments. The SoftBank Pepper social robot is designed for human-robot interaction and is commonly used in public spaces. The platform is undergoing security investigations. A dataset (DR) with normal records collected from Pepper's built-in autonomous interactive behavior is used to train the class model. The test set (TS) includes normal records and simulated attacks, such as LedsControl (remote control of robot's LEDs), JointsControl (remote control of robot's joints for specific poses), and WheelsControl (external control of robot movement via a remote joystick).

The logs include 256 real values from various sensors, excluding data from microphones and video cameras for privacy and simplicity in data processing. The training dataset (DR) consists of 14,144 records, with 9,900 for training and 4,244 for threshold evaluation. The test set (TS) contains 12,532 samples, with 8,288 abnormal samples distributed among different attack types: WheelsControl (30

2.4 Datasets preparation

Before the analysis we had to do some pre-processing on the datasets, all the operations are dataset-specific, since all three have different values. The only operation done equally on all datasets is the fourier analysis, explained later on.

2.4.1 Pepper

1. **Data Cleaning:**
    - Removes the first row of the DataFrame (`df`).
    - Removes near-constant columns (columns with the same value for most of the entries).
    - Drops columns that are entirely filled with NaN values.
    - Converts the 'timestamp' column to datetime format.

- Sets the 'timestamp' column as the index of the DataFrame.
- Drops the 'timestamp' column.

2. **Data Type Conversion:**

   - Converts all cell values in the DataFrame to float.

3. **Feature Selection:**

   - Calculates the variance for each column and keeps the 50 columns with the largest variances.
   - **Normal dataset before the cleaning**: *HeadYaw, HeadPicth, LShoulderPitch, LShoulderRoll, LElbowYaw, LElbowRoll, LWristYaw, LHand, RShoulderPitch, RShoulderRoll, RElbowYaw, RElbowRoll, RWristYaw, Rhand, HipRoll, HipPicth, KneePicth, WheelFLSpeed, WheelFRSpeed, WheelBSpeed, LaserR01X, LaserR01Y, LaserR02X, LaserR02Y, LaserR03X, LaserR03Y, LaserR04X, LaserR04Y, LaserR05X, LaserR05Y, LaserR06X, LaserR06Y, LaserR07X, LaserR07Y, LaserR08X, LaserR08Y, LaserR09X, LaserR09Y, LaserR10X, LaserR10Y, LaserR11X, LaserR11Y, LaserR12X, LaserR12Y, LaserR13X, LaserR13Y, LaserR14X, LaserR14Y, LaserR15X, LaserR15Y, LaserF01X, LaserF01Y, LaserF02X, LaserF02Y, LaserF03X, LaserF03Y, LaserF04X, LaserF04Y, LaserF05X, LaserF05Y, LaserF06X, LaserF06Y, LaserF07X, LaserF07Y, LaserF08X, LaserF08Y, LaserF09X, LaserF09Y, LaserF10X, LaserF10Y, LaserF11X, LaserF11Y, LaserF12X, LaserF12Y, LaserF13X, LaserF13Y, LaserF14X, LaserF14Y, LaserF15X, LaserF15Y, LaserF01X.1, LaserF01Y.1, LaserF02X.1, LaserF02Y.1, LaserF03X.1, LaserF03Y.1, LaserF04X.1, LaserF04Y.1, LaserF05X.1, LaserF05Y.1, LaserF06X.1, LaserF06Y.1, LaserF07X.1, LaserF07Y.1, LaserF08X.1, LaserF08Y.1, LaserF09X.1, LaserF09Y.1, LaserL10X, LaserL10Y, LaserL11X, LaserL11Y, LaserL12X, LaserL12Y, LaserL13X, LaserL13Y, LaserL14X, LaserL14Y, LaserL15X, LaserL15Y, LaserShovel01X, LaserShovel01Y, LaserShovel02X, LaserShovel02Y, LaserShovel03X, LaserShovel03Y, LaserVeR01X, LaserVeR01Y, LaserVeL01X, LaserVeL01Y, SonarF, SonarB, InfraRedL, InfraRedR, LedFaceRL1, LedFaceRL2, LedFaceRL3, LedFaceRL4, LedFaceRL5, LedFaceRL6, LedFaceRL7, LedFaceRL8, LedFaceGL1, LedFaceGL2, LedFaceGL3, LedFaceGL4, LedFaceGL5, LedFaceGL6, LedFaceGL7, LedFaceGL8, LedFaceBL1, LedFaceBL2, LedFaceBL3, LedFaceBL4, LedFaceBL5, LedFaceBL6, LedFaceBL7, LedFaceBL8, LedFaceRR1, LedFaceRR2, LedFaceRR3, LedFaceRR4, LedFaceRR5, LedFaceRR6, LedFaceRR7, LedFaceRR8, LedFaceGR1, LedFaceGR2, LedFaceGR3, LedFaceGR4, LedFaceGR5, LedFaceGR6, LedFaceGR7, LedFaceGR8, LedFaceBR1, LedFaceBR2, LedFaceBR3, LedFaceBR4, LedFaceBR5, LedFaceBR6, LedFaceBR7, LedFaceBR8, LedEarR1, LedEarR2, LedEarR3, LedEarR4, LedEarR5, LedEarR6, LedEarR7, LedEarR8, LedEarR9, LedEarR10, LedEarL1, LedEarL2, LedEarL3, LedEarL4, LedEarL5, LedEarL6,*

LedEarL7, LedEarL8, LedEarL9, LedEarL10, LedShoulderR, LedShoulderG, LedShoulderB, GyroscopeX, GyroscopeY, GyroscopeZ, AnglesX, AnglesY, AnglesZ, AccelerometerX, AccelerometerY, AccelerometerZ, TouchHeadF, TouchHeadR, TouchHeadM, TouchLHand, TouchRHand, ChestButton, BumperFR, BumperFL, BumperB, TabletTouchX, TabletTouchY, HeadYawStiff, HeadPicthStiff, LShoulderPitchStiff, LShoulderRollStiff, LElbowYawStiff, LElbowRollStiff, LWristYawStiff, LHandStiff, RShoulderPitchStiff, RShoulderRollStiff, RElbowYawStiff, RElbowRollStiff, RWristYawStiff, RhandStiff, HipRollStiff, HipPicthStiff, KneePicthStiff, WheelFLStiff, WheelFRStiff, WheelBStiff, HeadYawTemp, HeadPicthTemp, LShoulderPitchTemp, LShoulderRollTemp, LElbowYawTemp, LElbowRollTemp, LWristYawTemp, LHandTemp, RShoulderPitchTemp, RShoulderRollTemp, RElbowYawTemp, RElbowRollTemp, RWristYawTemp, RhandTemp, HipRollTemp, HipPicthTemp, KneePicthTemp, WheelFLTemp, WheelFRTemp, WheelBTemp

- **Normal dataset after the cleaning**: *WheelBStiff, WheelFRStiff, TabletTouchX, KneePicthTemp, TabletTouchY, HeadPicthTemp, HipPicthTemp, HipRollTemp, RElbowYawTemp, RhandTemp, RElbowRollTemp, RShoulderRollTemp, LElbowYawTemp, LHandTemp, WheelFRTemp, LShoulderRollTemp, WheelBTemp, HeadYawTemp, RShoulderPitchTemp, LShoulderPitchTemp, LaserShovel01X, LaserShovel02X, LWristYawTemp, LaserShovel03X, WheelFLTemp, RWristYawTemp, LaserR09X, LaserR07X, LaserR06X, LaserR08X, LaserF01X, LaserR05X, LaserR01X, LaserF02X, LaserF15X, LaserF14X, LaserR11X, LaserR12X, LElbowRollTemp, LaserR15X, LaserR10X, LaserL15X, AnglesZ, LaserShovel01Y, LaserShovel02Y, LaserF09X, LaserShovel03Y, LaserF13X, LaserF08X, LaserF07X*

4. **Sampling Rows:**
   - Calls a function `sample_rows(df, 100)` to perform the Fourier analysis with 100 as sample rate.
5. **Return DataFrame:**
   - Returns the modified DataFrame.
6. **Final Variables:**
   - **Normal:** *WheelBStiff, WheelFRStiff, TabletTouchX, KneePicthTemp, TabletTouchY, HeadPicthTemp, HipPicthTemp, HipRollTemp, RElbowYawTemp, RhandTemp, RElbowRollTemp, RShoulderRollTemp, LElbowYawTemp, LHandTemp, WheelFRTemp, LShoulderRollTemp, WheelBTemp, HeadYawTemp, RShoulderPitchTemp, LShoulderPitchTemp, LaserShovel01X, LaserShovel02X, LWristYawTemp, LaserShovel03X, WheelFLTemp, RWristYawTemp, LaserR09X, LaserR07X, LaserR06X, LaserR08X, LaserF01X, LaserR05X, LaserR01X, LaserF02X, LaserF15X, LaserF14X, LaserR11X, LaserR12X, LElbowRollTemp, LaserR15X,*

LaserR10X, LaserL15X, AnglesZ, LaserShovel01Y, LaserShovel02Y, LaserF09X, LaserShovel03Y, LaserF13X, LaserF08X, LaserF07X

- **Joint Control:** *KneePicthTemp, HipPicthTemp, RElbowRollTemp, HeadPicthTemp, RShoulderRollTemp, WheelFRTemp, WheelBTemp, LaserShovel01X, LaserShovel02X, LaserShovel03X, RElbowYawTemp, LaserR06X, LaserR07X, LaserF02X, HipRollTemp, LaserF01X, LaserR05X, LaserR08X, LElbowYawTemp, LaserR04X, LaserR15X, HeadYawTemp, LShoulderRollTemp, WheelFLTemp, LaserR02X, RShoulderPitchTemp, LShoulderPitchTemp, LaserR01X, LaserF07X, LaserL15X, LaserF15X, LaserR09X, LaserF14X, LaserF04X, LaserShovel01Y, LaserShovel02Y, LaserShovel03Y, LaserF05X, LaserR11X, LaserR12X, LaserF08X, LElbowRollTemp, LaserR10X, LaserF03X, LHandTemp, RWristYawTemp, LaserR01Y, RhandTemp, LaserL13X, LaserL15Y*
- **Wheels Control:** *HeadPicthTemp, LaserShovel01X, HipPicthTemp, LaserShovel02X, LaserShovel03X, KneePicthTemp, LShoulderRollTemp, RShoulderRollTemp, WheelFRTemp, LaserR09X, LaserR08X, LaserF08X, LaserF09X, LaserR10X, LaserF01X, LaserR06X, LElbowYawTemp, LaserR07X, LaserR11X, LaserF10X, LaserF06X, LaserF08X.1, LaserR01X, LaserF05X, LaserL12X, LaserF07X, LaserF09X.1, LaserL11X, AnglesZ, LaserF07X.1, LaserR15X, LaserL10X, WheelFLTemp, RElbowYawTemp, LaserL15X, LaserF01X.1, LaserR12X, LaserF04X, LaserR02X, LaserShovel01Y, LaserR05X, LaserR14X, LaserF02X, LaserR13X, LaserL13X, LaserF11X, LaserF03X.1, LaserShovel02Y, LaserF15X, LaserL14X*

2.4.2 Boat

1. **Data Cleaning:**
   - Checks if the first element of the first row is a string. If true:
     - Removes columns with string values.
     - Drops the first row of the DataFrame.
     - Writes the modified DataFrame back to the original CSV file.
   - Drops columns that are entirely filled with NaN values.
2. **Column Removal:**
   - Drops the 'net_rec_tstamp' column if it exists.
3. **Timestamp Conversion:**
   - Converts the 'net_send_tstamp' column to datetime format.
   - Sets the 'net_send_tstamp' column as the index of the DataFrame.
4. **Data Type Conversion:**
   - Converts all cell values in the DataFrame to float.

5. **Sampling Rows:**
   - Calls a function `sample_rows(df, 5)` to perform the Fourier analysis with 5 as sample rate.
6. **Return DataFrame:**
   - Returns the modified DataFrame.
7. **Final Variables**
   - **Normal:** *gps_speed, motor_vel_1, proximity, net_rec_offset, game_rotation_vector, accelerometer_uncalibrated, gps_bearing, accelerometer, net_send_length, water_oxygen, gyroscope_uncalibrated, gps_accuracy, step_counter, step_detector, gps_longitude, gyroscope, gps_latitude, magnetic_field, gravity, sensor, rotation_vector, net_rec_malformed, linear_acceleration, gps_altitude, light, orientation, gps_time, net_rec_port, net_rec_length, motor_vel_0, magnetic_field_uncalibrated, pressure, geomagnetic_rotation _vector.*
   - **Faul Stucked:** *accelerometer, accelerometer_uncalibrated, game_rotation_vector, geomagnetic_rotation_vector, gps_accuracy, gps_altitude, gps_bearing, gps_latitude, gps_longitude, gps_speed, gps_time, gravity, gyroscope, gyroscope_uncalibrated, light, linear_acceleration, magnetic_field, magnetic_field_uncalibrated, net_rec_length, net_rec_malformed, net_rec_offset, net_rec_port, net_send_length, orientation, pressure, proximity, rotation_vector, sensor, step_counter, step_detector, motor_vel_0, motor_vel_1, water_oxygen.*
   - **Fault GPSDown:** *gps_speed, motor_vel_1, proximity, net_rec_offset, game_rotation_vector, accelerometer_uncalibrated, gps_bearing, accelerometer, net_send_length, water_oxygen, gyroscope_uncalibrated, gps_accuracy, step_counter, step_detector, gps_longitude, gyroscope, gps_latitude, magnetic_field, gravity, sensor, rotation_vector, net_rec_malformed, linear_acceleration, gps_altitude, light, orientation, gps_time, net_rec_port, net_rec_length, motor_vel_0, magnetic_field_uncalibrated, pressure, geomagnetic_rotation_vector.*

2.4.3 Swat

1. **Data Cleaning:**
   - Takes the last `SUBSET_SIZE` rows of the DataFrame.
   - Removes near-constant columns.
   - Drops columns that are entirely filled with NaN values.
   - Converts the 'Timestamp' column to datetime format.
   - Sets the 'Timestamp' column as the index of the DataFrame.
   - Drops specified columns ('Timestamp', 'Normal/Attack').
2. **Sampling Rows:**

   - Calls a function `sample_rows(df, 50)` to perform the Fourier analysis with 50 as sample rate.

3. **Return DataFrame:**

   - Returns the modified DataFrame.

4. **Final variables**

   - **Normal:** *FIT101, LIT101, MV101, P101, AIT201, AIT202, AIT203, FIT201, MV201, P203, P205, DPIT301, FIT301, LIT301, MV301, MV302, MV303, MV304, P301, P302, AIT401, AIT402, FIT401, LIT401, P402, UV401, AIT501, AIT502, AIT503, AIT504, FIT501, FIT502, FIT503, FIT504, P501, PIT501, PIT502, PIT503, FIT601, P602*
   - **Attack:** *FIT101, LIT101, MV101, P101, P102, AIT201, AIT202, AIT203, FIT201, MV201, P201, P203, P204, P205, P206, DPIT301, FIT301, LIT301, MV301, MV302, MV303, MV304, P302, AIT401, AIT402, FIT401, LIT401, P402, P403, UV401, AIT501, AIT502, AIT503, AIT504, FIT501, FIT502, FIT503, FIT504, P501, PIT501, PIT502, PIT503, FIT601, P602*

2.4.4 Fourier analysis

In our analysis, we applied **Fourier** analysis to each column, so to each variable, of our datasets.

This technique decomposes complex signals into sinusoidal components, allowing us to extract the underlying frequency information. For each column, we identified the top 5 components with the highest amplitudes, focusing on the most significant features of the data. Subsequently, we sampled the signal at a multiple of the frequency associated with these dominant components. This step was taken to simplify the data representation and potentially reduce dimensionality for further analysis.

More precisely, we multiplied our max frequencies by these factors for each dataset:

- **Pepper**: 100
- **Boat**: 5
- **Swat**: 5000

The differences of the sampling rate is due to the different frequencies of each signal, if the frequence is too small then we have to multiply it for an higher number in order to obtain an integer.

3 Experiments

To perform offline anomaly detection with causal discovery we used the **PCMCI** algorithm, with both **PARCORR** and **KNN**. PCMCI (partial correlation and mutual information-based causal inference) is a framework that combines partial correlation and mutual information to infer causal relationships between variables. The idea is to use partial correlation to measure strength of linear relationships between variables while considering the influence of other variables. Mutual information is also employed to capture non-linear dependencies between variables. The PCMCI algorithm involves constructing a graph, often referred as causality graph, where nodes represent variables, and edges represent inferred causal relationships. The strength and direction of the edges are determined based on the partial correlation and mutual information values.

- **PARCORR**: Parcorr stands for partial correlation and it's a function used for computing partial correlation coefficients between time series variables. The partial correlation coefficient between two variables measures the strength and direction of their linear relationship while controlling for the influence of other variables in the dataset. It quantifies the association between two variables after removing the linear effect of all other variables.
- **KNN**: KNN is a machine learning algorithm, in the context of causal discovery it can be employed to enhance the identification of causal relationships between variables. The idea is to leverage the concept of proximity in feature space, it wants to obtain complex non-linear dependencies. KNN, used with PCMCI, considers the local structure of the data. KNN identifies the nearest neighbors of each data point in the feature space, and the relationships between variables can be assessed by taking into account the values of their neighbors. This allows the algorithm to capture non-linear dependencies that might be missed by traditional statistical measures.

3.1 Hyperparameters

To perform these experiments there were some hyperparameters to set.

- **tau max**: represents the maximum time lag or delay that the algorithm consider when examining the potential causal relationship between variables, it determines the maximum number of time steps by which one variable might influence another. In other words the algorithm will assess the relationship between variables at lags from 1 to tau max.
- **tau min**: represents the minimum time lag considered in the analysis. It sets lower limit for the range of time lags examined. Relationships with time lags smaller than tau min are not considered during time analysis.
- **pc alpha**: is the significance level or threshold used for conditional independence tests. This parameter determines the threshold for considering variables as conditionally independent.

- **alpha**: is the significance level for the causal graph.

4 Results

We did different analysis on all datasets, changing parameters such as **pc alpha**, **alpha**, and the number of neighbours in the case of **knn**, to see how the analysis would change. Plot legends:

- **Cross-MCI** (for edges): refers to the links in the causal grah, representing cross-variable interactions. In this context it indicate causal relationships between different variables in the dataset, it usually represent direct causal influences from one variable to another.
- **Auto-MCI** (for nodes): refers to autocorrelation value, represent the temporal dependecies of a variable on its past values.

If we put "verbosity" = 1, the function that performs the algorithm plots useful information, which are basically the text version of the causal graph. We decided to include only the plots since are more representative, but we will still explain the outputs of "verbosity" = 1:

- **p-val**: the p-value is a probability that measures the evidence against the null hypotesys (baseline assumption, in causal discovery revolves around the absence of a causal relationship between variables), the smaller it is the more the two variables are correlated, if its higher that means that there is not enough evidence to reject the hypotesys of no effect or no relationship.
- **val**: represents the value associated with the strenght and direction of the relationship between the two variables, a negative correlation means that as one variable increases the other one decreases, if its positive means that as one variable increases the other one also increases. The magnitude (abs of value) indicates the strenght of the relationship.
- **Causal link**: if its unoriented it means that the discovery relationship between the variables is unoriented or undirected. In causal discovery, orientation enstabilish the direction of causality. An unoriented link implies that the algorithm has identified a relationship between the variables but has not determined the direction of causation.
- **Lag**: each variable has a list of correlations (if present), and each correlation indicates also the lag value, which is the "lagged time" in which the correlation occurs, in the plots the lag is indicated over the causal link.

4.1 Swat

- Swat, normal, linear, parcorr, pcalpha=0.01: link to image

- Swat, normal, linear, parcorr, pcalpha=0.05: link to image
- Swat, attack, linear, parcorr, pcalpha=0.01: link to image
- Swat, attack, linear, parcorr, pcalpha=0.05: link to image
- Swat, normal, not linear, knn=10, pcalpha=0.05: link to image
- Swat, normal, not linear, knn=10, pcalpha=0.03: link to image
- Swat, attack, not linear, knn=10, pcalpha=0.05: link to image
- Swat, attack, not linear, knn=10, pcalpha=0.03: link to image

4.2 Boat

- Boat, normal, linear, parcorr, pcalpha=0.01: link to image
- Boat, normal, linear, parcorr, pcalpha=0.03: link to image
- Boat, normal, linear, parcorr, pcalpha=0.05: link to image
- Boat, gpsdown, linear, parcorr, pcalpha=0.01: link to image
- Boat, gpsdown, linear, parcorr, pcalpha=0.03: link to image
- Boat, gpsdown, linear, parcorr, pcalpha=0.05: link to image
- Boat, stucked, linear, parcorr, pcalpha=0.01: link to image
- Boat, stucked, linear, parcorr, pcalpha=0.03: link to image
- Boat, stucked, linear, parcorr, pcalpha=0.05: link to image
- Boat, normal, not linear, knn=10, pcalpha=0.03: link to image
- Boat, normal, not linear, knn=3, pcalpha=0.05: link to image
- Boat, normal, not linear, knn=5, pcalpha=0.05: link to image
- Boat, normal, not linear, knn=10, pcalpha=0.05: link to image
- Boat, gpsdown, not linear, knn=5, pcalpha=0.03: link to image
- Boat, gpsdown, not linear, knn=10, pcalpha=0.03: link to image
- Boat, gpsodwn, not linear, knn=3, pcalpha=0.05: link to image
- Boat, gpsdown, not linear, knn=5, pcalpha=0.05: link to image
- Boat, gpsdown, not linear, knn=10, pcalpha=0.05: link to image
- Boat, stucked, not linear, knn=5, pcalpha=0.03: link to image
- Boat, stucked, not linear, knn=10, pcalpha=0.03: link to image

- Boat, stucked, not linear, knn=3, pcalpha=0.05: link to image
- Boat, stucked, not linear, knn=5, pcalpha=0.05: link to image
- Boat, stucked, not linear, knn=10, pcalpha=0.05: link to image

4.3 Pepper

- Pepper, normal, linear, parcorr, pcalpha=0.01: link to image
- Pepper, normal, linear, parcorr, pcalpha=0.03: link to image
- Pepper, normal, linear, parcorr, pcalpha=0.05: link to image
- Pepper, normal, linear, parcorr, pcalpha=0.05, alpha=0.000005: link to image
- Pepper, wheels, linear, parcorr, pcalpha=0.01: link to image
- Pepper, wheels, linear, parcorr, pcalpha=0.03: link to image
- Pepper, wheels, linear, parcorr, pcalpha=0.05: link to image
- Pepper, wheels, linear, parcorr, pcalpha=0.05, alpha=0.00005: link to image
- Pepper, joint, linear, parcorr, pcalpha=0.01: link to image
- Pepper, joint, linear, parcorr, pcalpha=0.03: link to image
- Pepper, joint, linear, parcorr, pcalpha=0.05: link to image
- Pepper, normal, not linear, knn=10, pcalpha=0.03: link to image
- Pepper, normal, not linear, knn=10, pcalpha=0.05: link to image
- Pepper, wheels, not linear, knn=10, pcalpha=0.03: link to image
- Pepper, wheels, not linear, knn=10, pcalpha=0.05: link to image
- Pepper, joint, not linear, knn=10, pcalpha=0.03: link to image
- Pepper, joint, not linear, knn=10, pcalpha=0.05: link to image