

Lecture „Intelligent Systems“

Chapter 12: Mutual Influences

Prof. Dr.-Ing. habil. Sven Tomforde / Intelligent Systems
Winter term 2020/2021

Contents

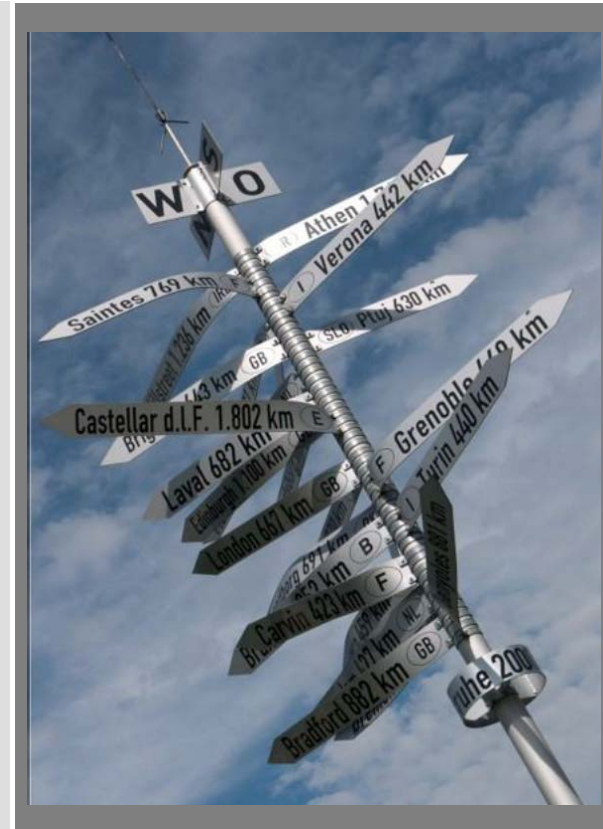
- Motivation
- Running example and approach
- Detecting influences and dependencies
- Behaviour of measures in test cases
- Influence detection at runtime
- Influence-aware self-adaptation
- Conclusion
- Further readings

Goals

Students should be able to:

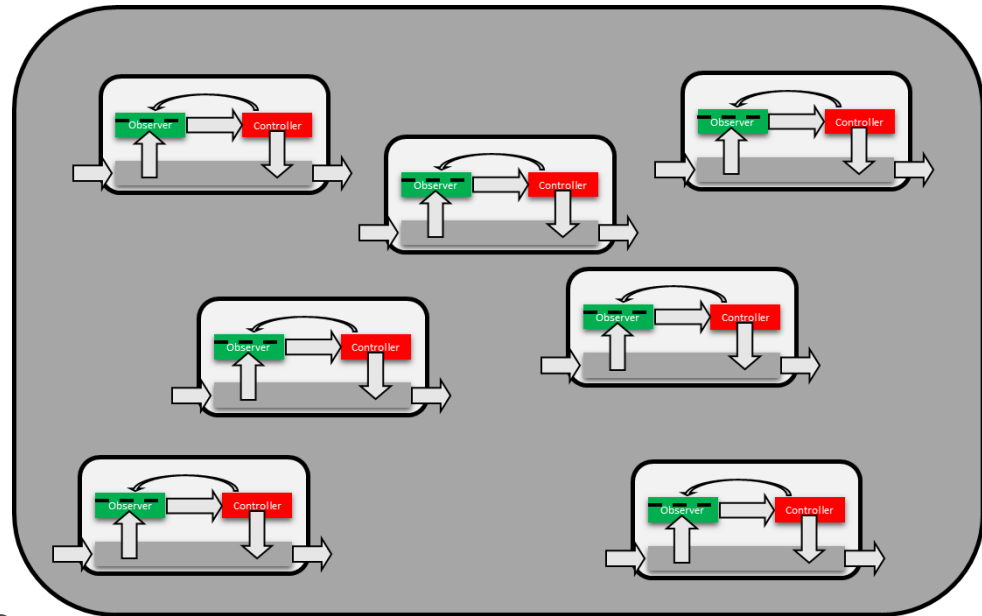
- define and motivate mutual information detection
- characterise and compare the different dependency measures
- explain the methodology for mutual influence detection
- apply the detected mutual influence information to self-learning adaptation strategies

- Motivation
- Running example and approach
- Detecting influences and dependencies
- Behaviour of measures in test cases
- Influence detection at runtime
- Influence-aware self-adaptation
- Conclusion and further readings



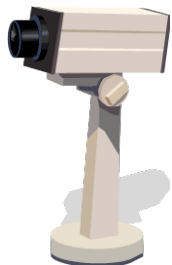
System of Systems

- Subsystems act autonomously
- E.g. adapt their configuration depending on changes in the environmental conditions
- Means: The productive part really does something – that in turn manipulates the environmental state
- **Shared environment:**
 - Actions have an impact on the situations observed by other subsystems
 - Actions influence the utility of others



Smart Camera Network

- Each camera act autonomously
- Decides about its own pan/tilt/zoom configuration



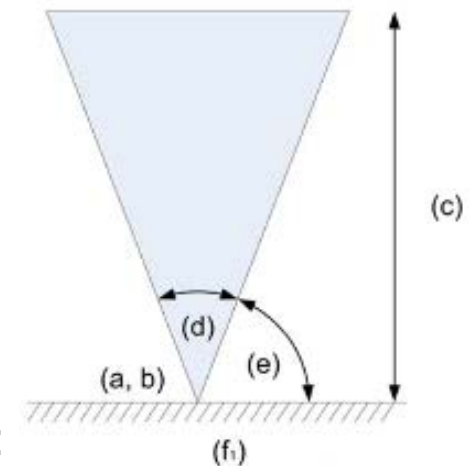
$[\text{Pan}_1, \text{Tilt}_1, \text{Zoom}_1]$
Performance₁



$[\text{Pan}_2, \text{Tilt}_2, \text{Zoom}_2]$
Performance₂

Goals:

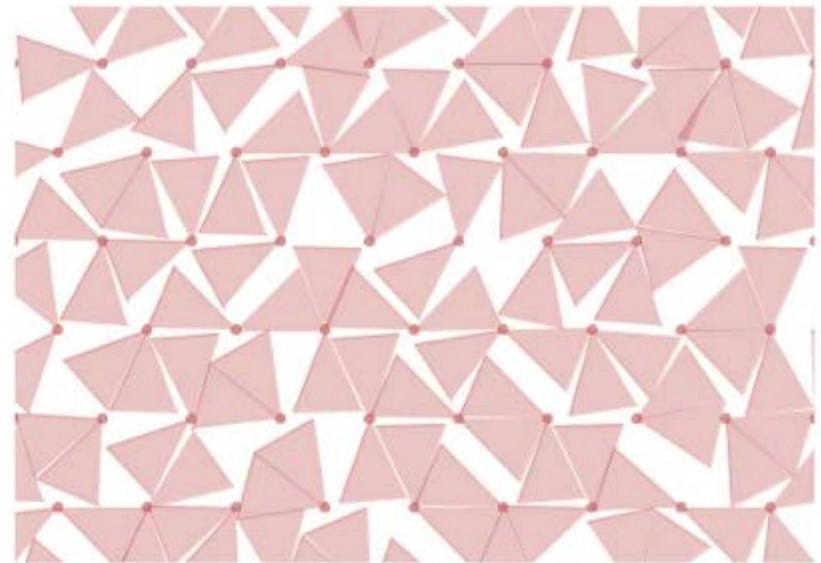
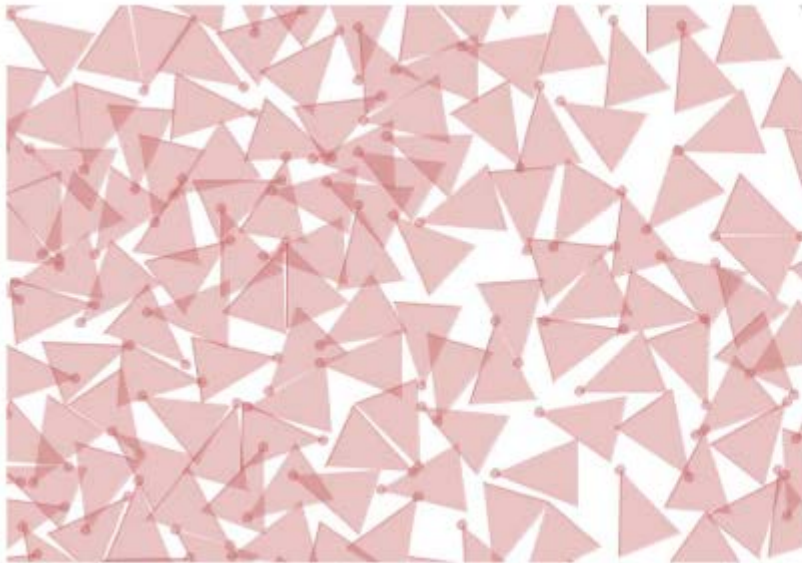
- Coverage optimisation
- Tracking of objects
- 3D reconstruction
- ...



Abstract camera model and configuration parameters:

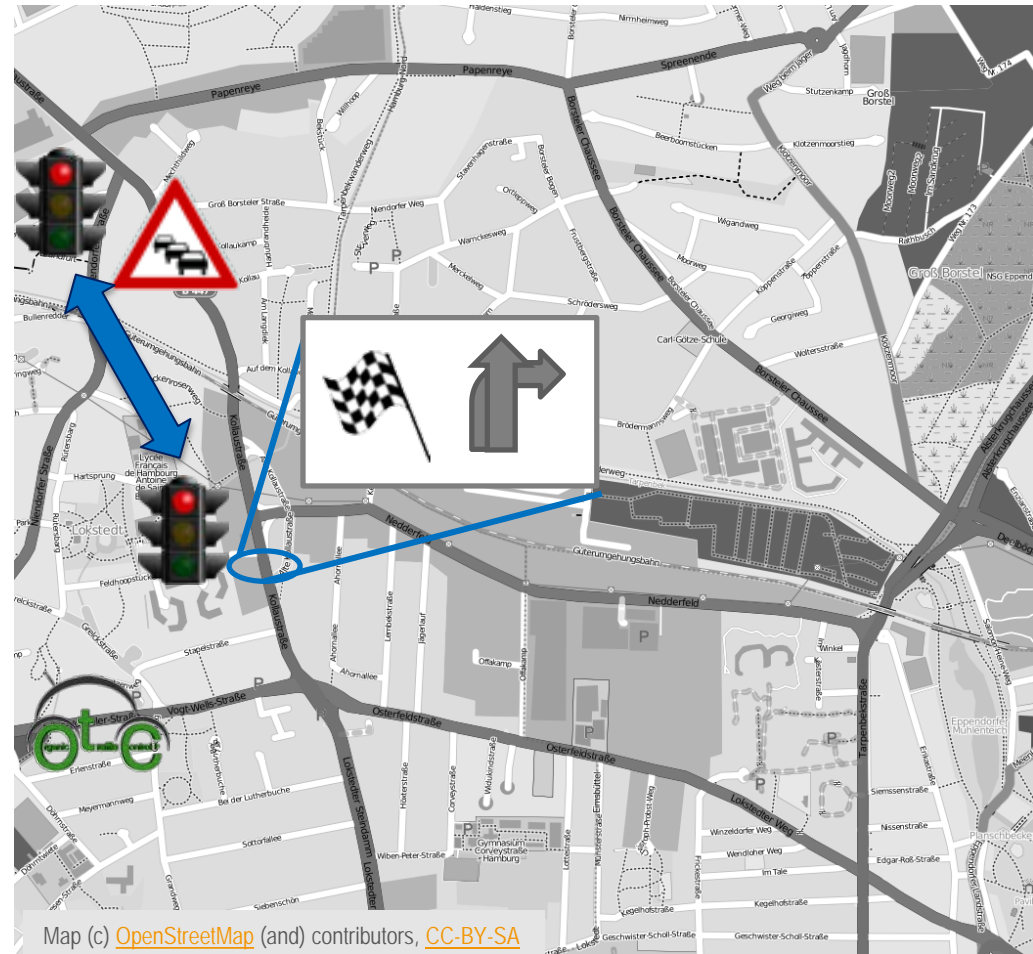
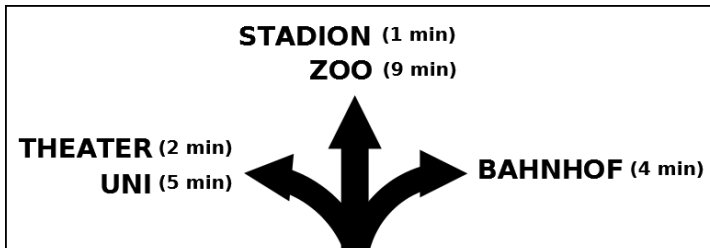
Coverage optimisation

- Left: Each camera acts isolated
- Right: Each camera acts autonomously but considers the others' configuration



Traffic management

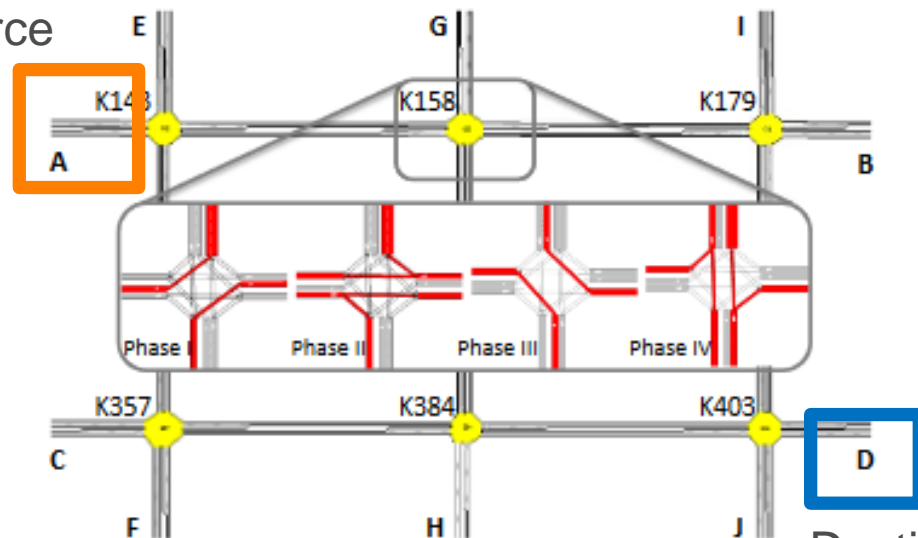
- Dynamic route guidance
- Each intersection decides autonomously where to route traffic streams
- Basis: Local traffic conditions
- Actions: Control variable message sign



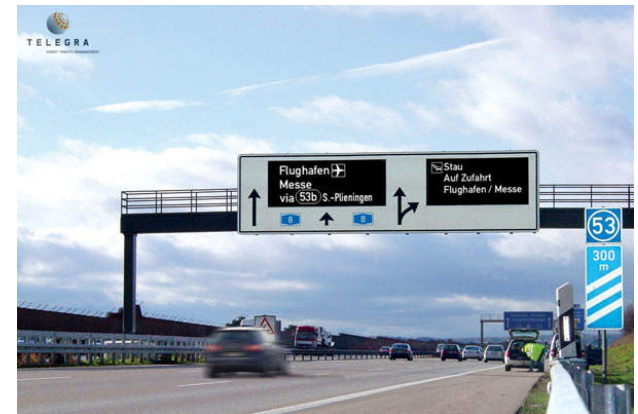
Route guidance


- Each action of an intersection controller impacts the conditions of upstream intersections
- Consequence: Change in the utility

Source



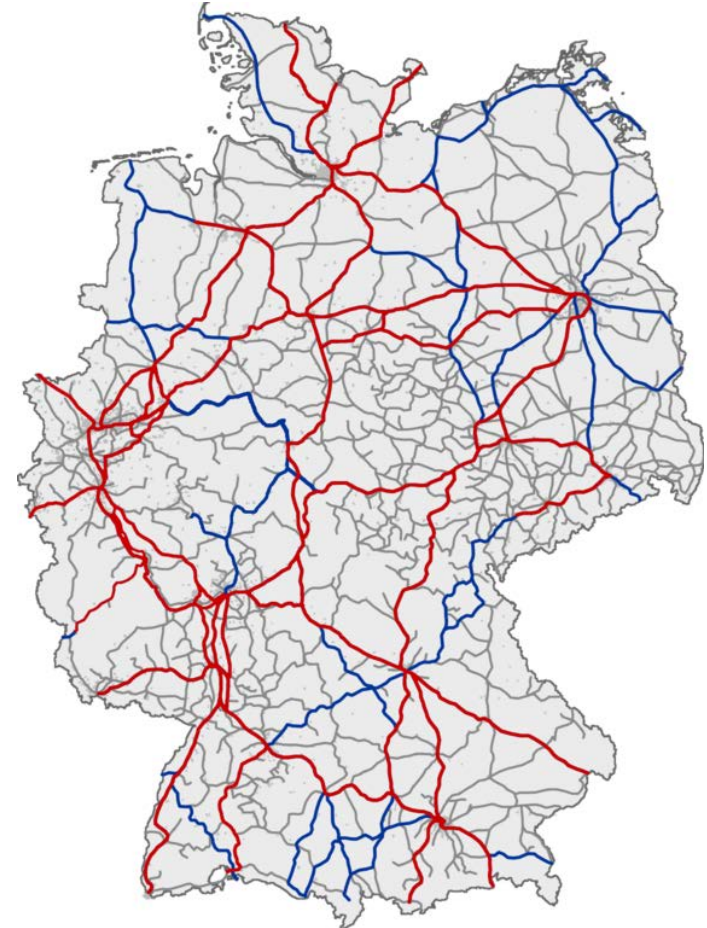
Destination



- 
- Largest blackout in the USA
 - 55m households
 - Single power plant shut down
 - Failure of several voltage transmission lines
 - Reasons
 - Network utilisation at shutdown time
 - Supervisory Control and Data Acquisition introduced vulnerabilities

Smart grids

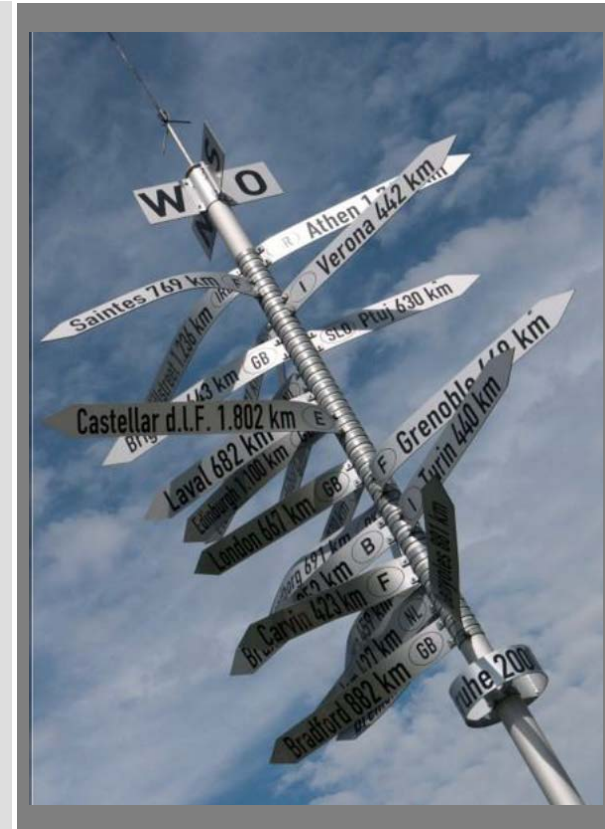
- Before: Centralised
 - Many small producers
 - Dynamic demand schemes
 - Now: Autonomous entities
 - System stability at e.g. 50Hz
 - Communication based on ICT
- ICT depends on available power,
the smart grid on available ICT
- Similar examples: railway system,
etc.



Mutual Influence (MI)

Mutual Influences are effects of configurations of other subsystems in a shared environment on the utility achievement of an individual subsystem in a particular situation. Typically, this means that a specific configuration of another subsystem or a group of subsystems negatively influences the gathered reward.

- Motivation
- Running example and approach
- Detecting influences and dependencies
- Behaviour of measures in test cases
- Influence detection at runtime
- Influence-aware self-adaptation
- Conclusion and further readings



Collaborative Cookie Manipulation

- Two robots want to lift a cookie, but one alone cannot lift it.
- Both have the possibility to configure their arms as UP or DOWN
- Characteristics
 - If both do UP, the cookie is lifted
 - Then: each robot gets a reward of 1
 - Otherwise: Not lifted and reward 0
- The procedure is repeated in each step



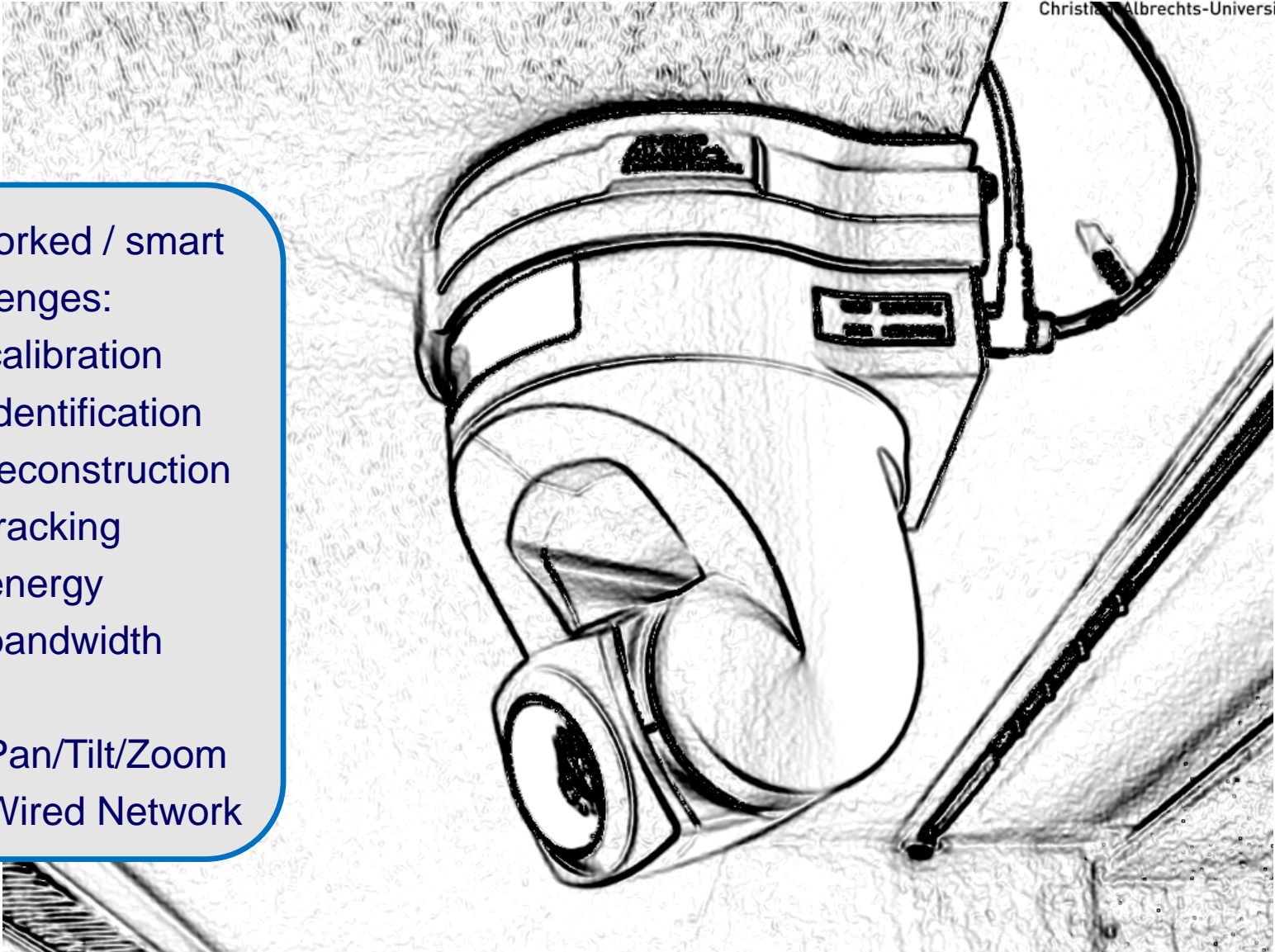
Two-man crosscut saw

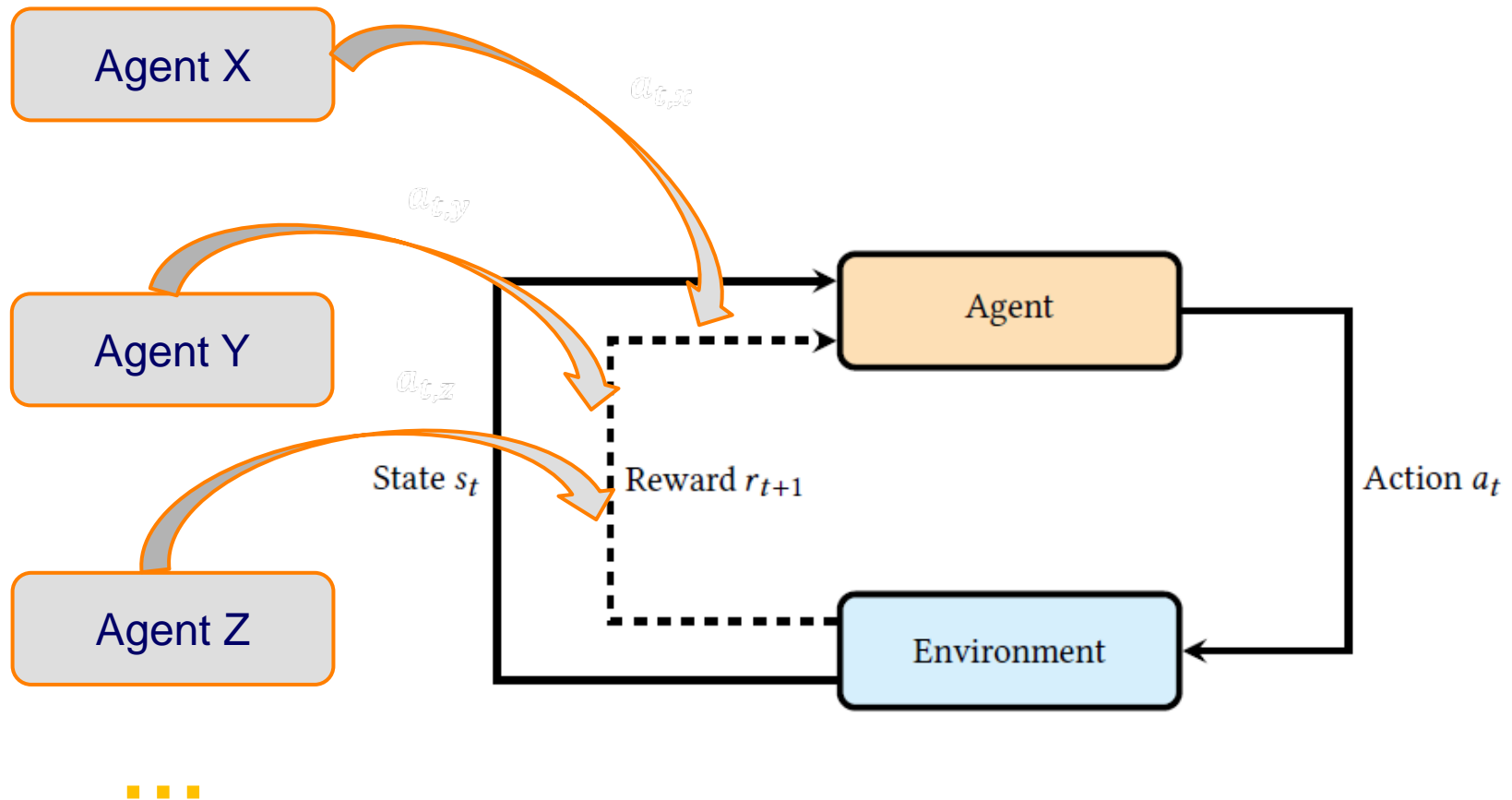
- Imagine two robots with a crosscut saw
- Each can PUSH or PULL
- Characteristics:
 - If one PUSHes and the other one PULLs, the saw moves and they get a reward of 1
 - Otherwise: Nothing happens and they get a reward of 0



Example 3: Smart Camera Networks

- Networked / smart
- Challenges:
 - calibration
 - identification
 - reconstruction
 - tracking
 - energy
 - bandwidth
- Here:
 - Pan/Tilt/Zoom
 - Wired Network





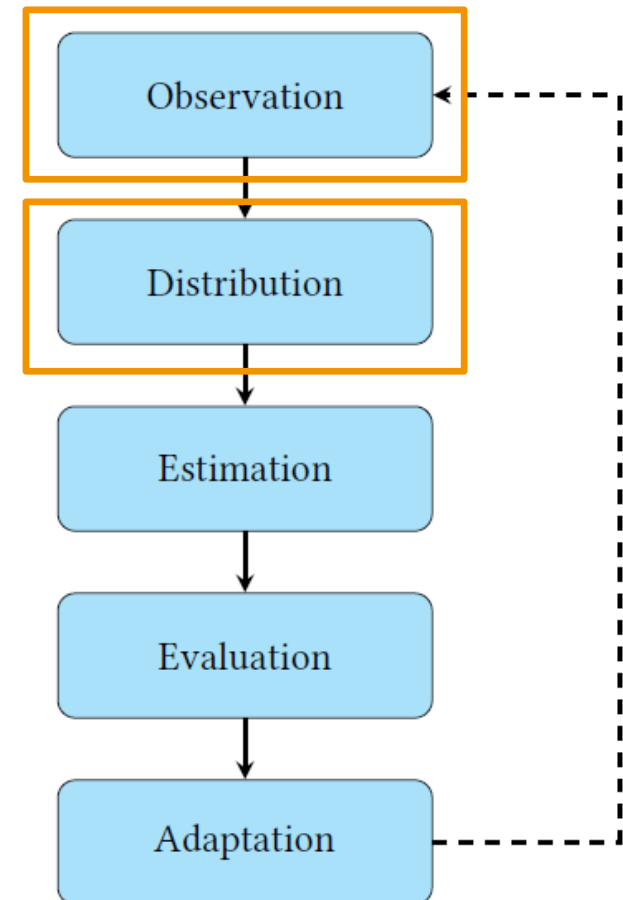
Mutual influence detection

1.Observation

- Continuously observe the configuration
- Estimate the current utility
- E.g. the pan, tilt, and zoom of a camera and the corresponding reward in terms of the given goals.
- These observations will typically be done by the system itself, but they could as well come from an external entity.

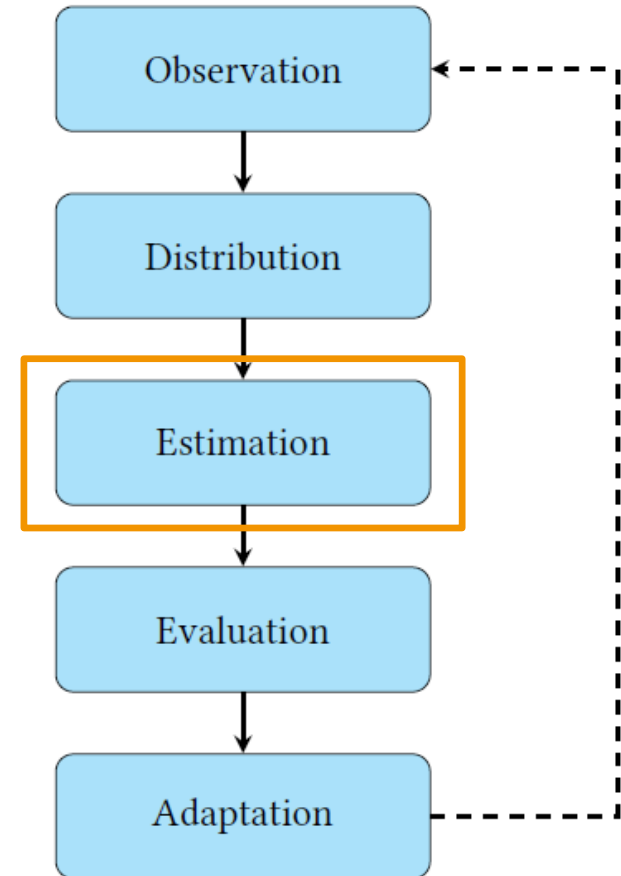
2.Distribution

- Gather configurations from other systems
- Provide your own one to them



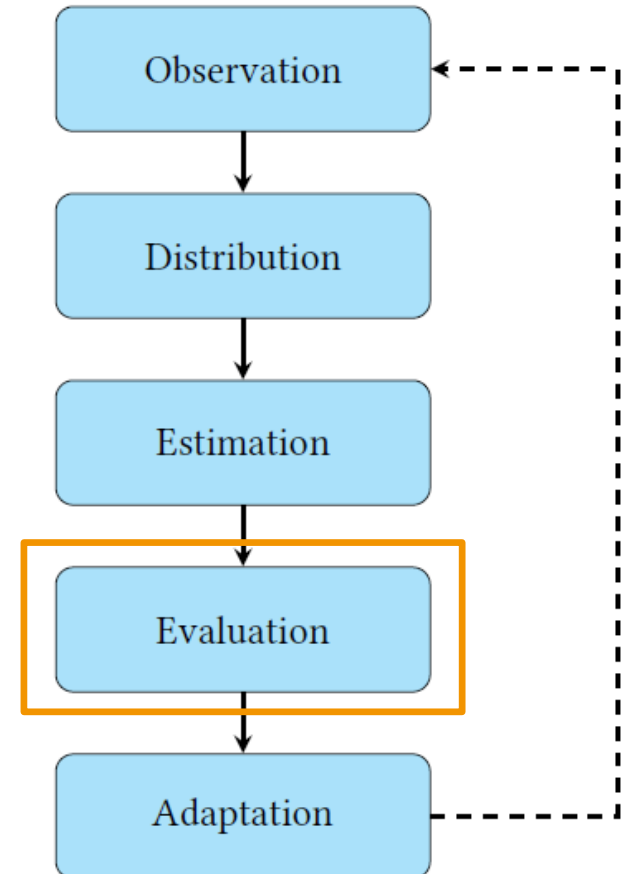
3. Estimation

- Estimate the dependency value by relating the own reward to the configurations of the other systems.
- Use of stochastic dependency measures that estimate associations between the reward of a system A and the configuration components of a second system B.
- Dependency measures identify correlations between two random variables X and Y.
- Reward of A is identified with a random variable X and the configuration of entity B with a random variable Y.
- Mapping implies that if the association between X and Y is high, we also have a high influence of B on A since it reflects that the configurations of B matter for the reward of A
- Vice versa, if the association is low, we do not see an influence



4. Evaluation

- Compare the values calculated for the different (other) systems and their configuration components
- **Approach A:** Compare the influence values to a fixed threshold
 - Can be difficult since an appropriate threshold might not be available for each application
- **Approach B:** Influence values can be compared on a relative basis between the systems
 - Leads to a ranking of systems according to their influence
- **Approach C:** Simulate an independent system with the same configuration space and compare “artificial” vs. “real” values
 - Allows us to decide on a basis that only includes one other system

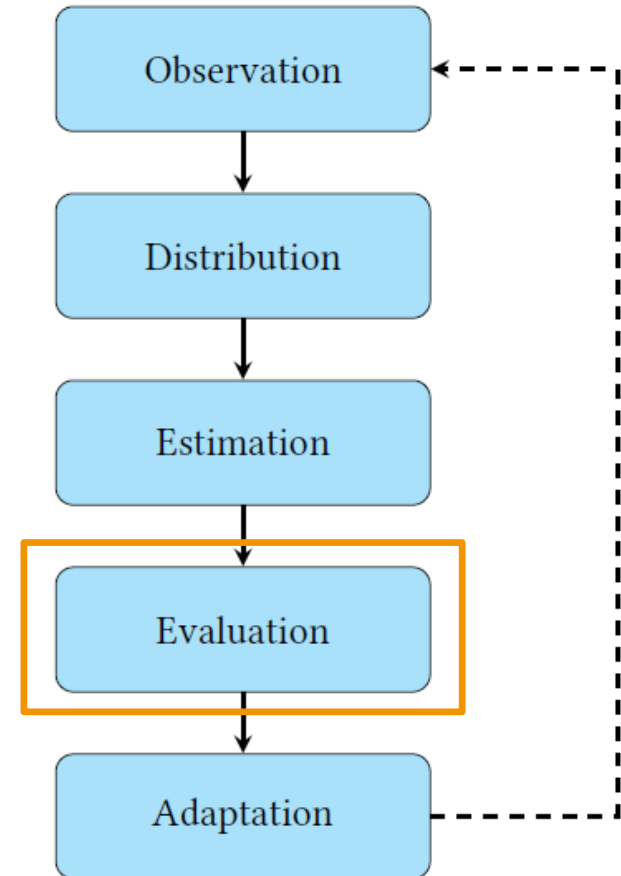


Regarding the evaluation:

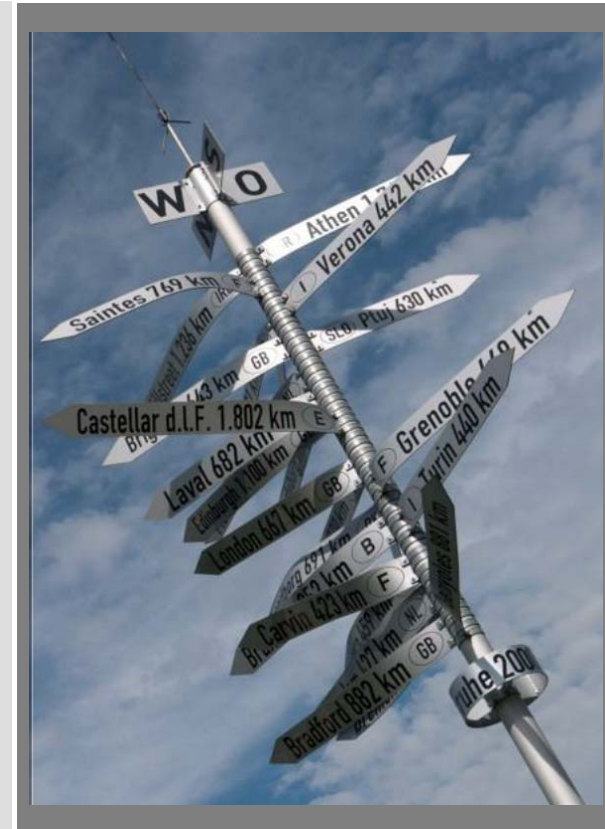
- What are the major advantages/disadvantages of the three approaches?

5. Adaptation

- Addresses the influences in the control strategy
- Can take various forms:
 - Especially when applied at design-time, the designer can decide on a case-by-case basis.
 - For a self-adaptive solution, a learning algorithm includes the configuration of the influencing systems in the situation description during runtime.



- Motivation
- Running example and approach
- **Detecting influences and dependencies**
- Behaviour of measures in test cases
- Influence detection at runtime
- Influence-aware self-adaptation
- Conclusion and further readings



Measure 1: Pearson Correlation

- Sometimes just “correlation coefficient”
- Advantages:
 - Simple implementation
 - Fast calculation
- Drawbacks:
 - Only linear correlations can be detected
 - Means: It can fail in case of more complex dependencies
 - Calculates the distance between realisations of the random variable which might make it not well suited for some problems.
- Measure
 - Assumes values between -1 and 1
 - -1 indicates a perfect negative linear correlation and 1 a perfect positive correlation. 0 means that there is no linear correlation
 - Uses absolute values for comparison when considering the influence of different systems

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

x_i and y_i : Gathered examples (e.g. configuration and reward)

\bar{x} and \bar{y} : Mean values

Measure 2: Kendall Rank

- Calculates ranks of gathered samples
- Advantages:
 - Can be computed rather fast
 - Can detect monotone dependencies (better than linear)
- Drawbacks:
 - Not sufficient for all applications
- Measure:
 - Assumes values between -1 and 1
 - -1 indicates a perfect monotone declining relationship
 - 1 a perfect monotone increasing relationship
 - Independent variables: a value around 0 should be considered

$$\tau = \frac{(\# \text{ concordant Pairs}) - (\# \text{ discordant Pairs})}{n(n-1)/2}$$

Concordant pair: two samples (x_i, y_i) , (x_j, y_j) where the ranks of the elements agree

If $x_i > x_j$ then $y_i > y_j$ or

If $x_i < x_j$ then $y_i < y_j$

Cases with $x_i = x_j$ and $y_i = y_j$ are ignored

Measure 3: Spearman Rank

- Similar to Kendall
- Measure:
 - Ranks of the samples are calculated
 - Instead of a comparison between the samples, the Pearson correlation coefficient is calculated on the ranks.
 - Means: values range from -1 to 1
 - Value of 0 means that no dependency has been detected

$$\rho = \frac{\sum_{i=1}^n (rg(x_i) - \overline{rg_x})(rg(y_i) - \overline{rg_y})}{\sqrt{\sum_{i=1}^n (rg(x_i) - \overline{rg_x})^2} \sqrt{\sum_{i=1}^n (rg(y_i) - \overline{rg_y})^2}}$$

x_i and y_i : Gathered examples (e.g. configuration and reward)

n : number of samples

$rg(x)$: Rank of sample x

$\overline{rg_x}$: Averaged rank of sample x_i

Measure 4: Distance covariance

- Extension to the Pearson correlation
- Takes the distance between the samples into account
- Advantages:
 - Can find different types of dependencies
- Drawbacks:
 - Not suitable for online calculation
 - Reason: distances have to be calculated for the entire sample set

$$\frac{1}{n^2} \sum_{j=1}^n \sum_{k=1}^n A_{j,k} B_{j,k}$$

$$a_{j,k} = \|x_j - x_k\| \text{ and } b_{j,k} = \|y_j - y_k\|$$

$$A_{j,k} := a_{j,k} - \bar{a}_{j\cdot} - \bar{a}_{\cdot k} + \bar{a}_{\cdot\cdot}$$

$\bar{a}_{j\cdot}$: Mean of row j

$\bar{a}_{\cdot k}$: Means of column k

$\bar{a}_{\cdot\cdot}$: Mean of entire matrix

Measure 5: Mutual information

- Shannon, Information Theory
- Advantages:
 - Online calculation possible
- Drawbacks:
 - Maximal possible value depends on the structure of the random variables
 - I.e. values can be normalised, but comparability is limited
- Measure:
 - Probabilities can be easily approximated using a frequency counting

$$I(X; Y) := \sum_{x \in X} \sum_{y \in Y} p(x, y) \text{ld} \left(\frac{p(x, y)}{p(x)p(y)} \right)$$

$p(x, y)$: joint probability of the events x and y

$p(x)$ and $p(y)$: are the marginal probabilities of x and y

Measure 6: Maximal information

- Extension of MI for real-values
- Advantages:
 - Automatically calculates the binning that results in the maximal MI for a given data set
- Drawbacks:
 - Not an online method
- Measure:
 - Denominator serves for normalisation of the value
 - Computationally expensive to determine the values for all possible binnings
 - There are heuristics available

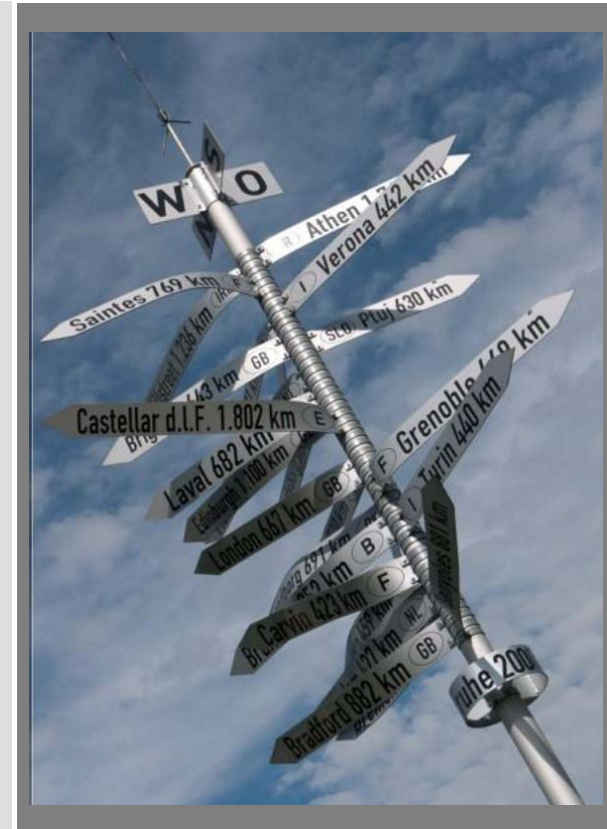
$$MIC(X, Y) := \max_{n_x n_y < B} \frac{I(X; Y)}{\log(\min(n_x, n_y))}$$

n_x and n_y : Number of bins for x and y

$I(X; Y)$: Mutual Information (MI)

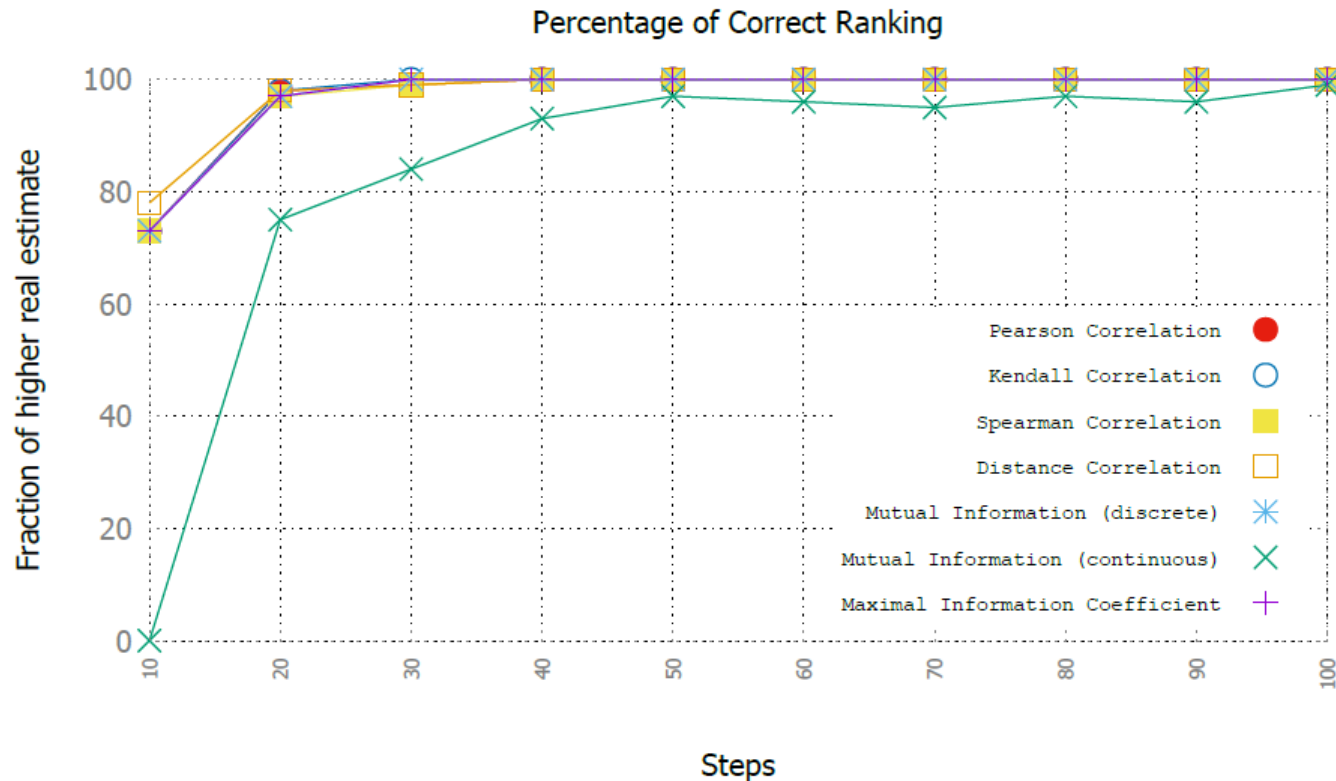
Number of bins is limited by a threshold B that is by default determined depending on the sample size

- Motivation
- Running example and approach
- Detecting influences and dependencies
- **Behaviour of measures in test cases**
- Influence detection at runtime
- Influence-aware self-adaptation
- Conclusion and further readings



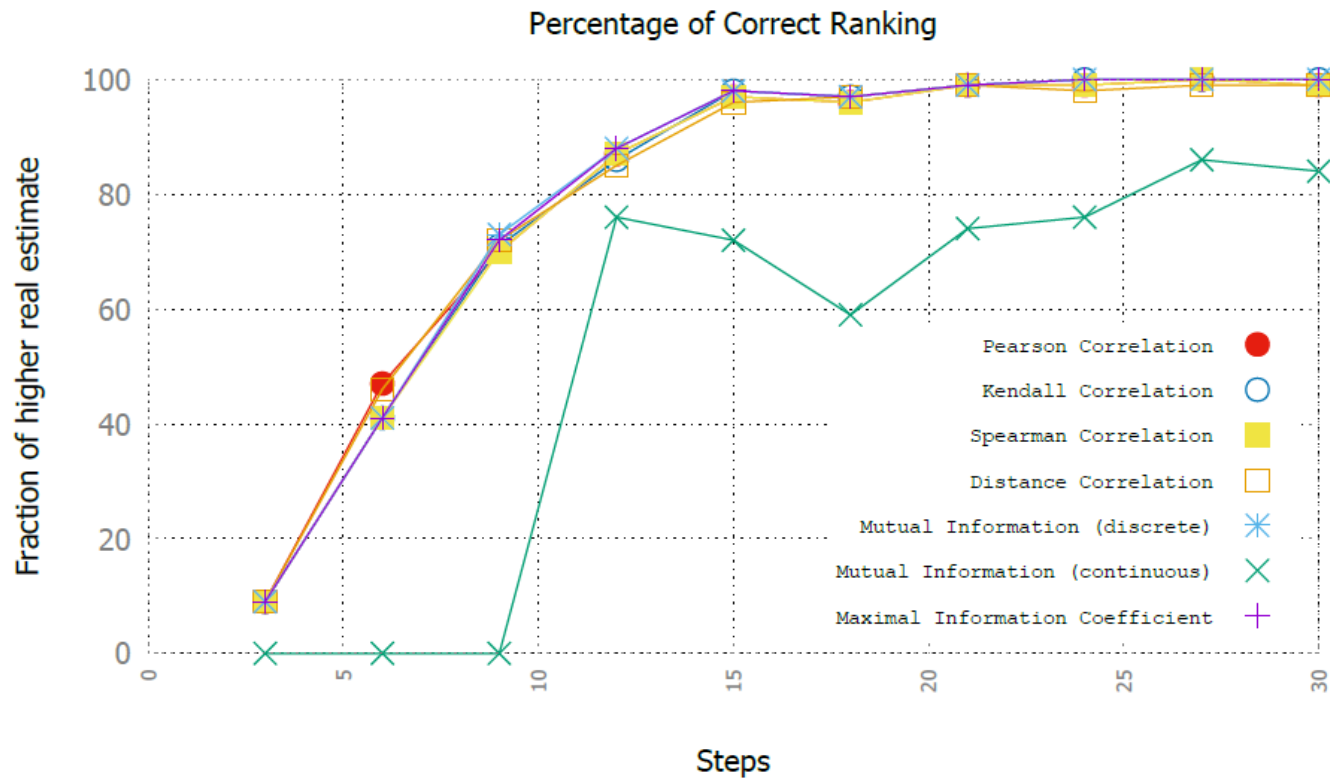
Evaluation of measures for collaborative cookie scenario

- First 100 steps



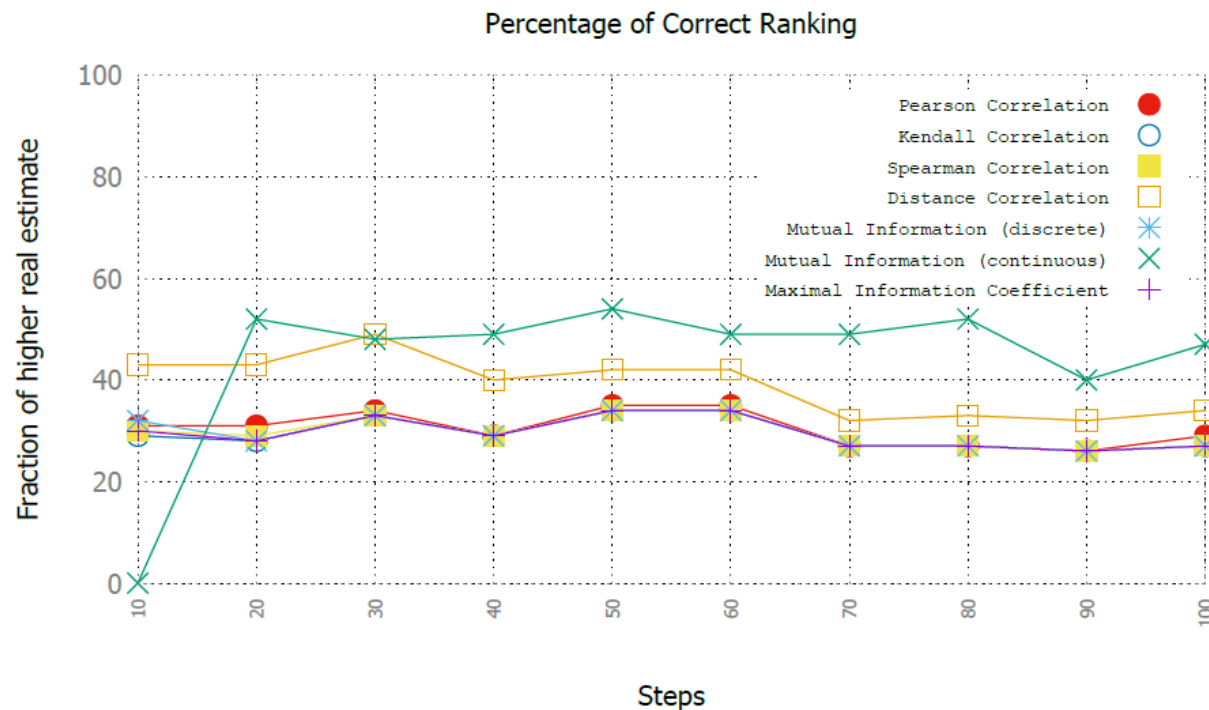
Evaluation of measures for collaborative cookie scenario

- Detailed view of first 30 steps



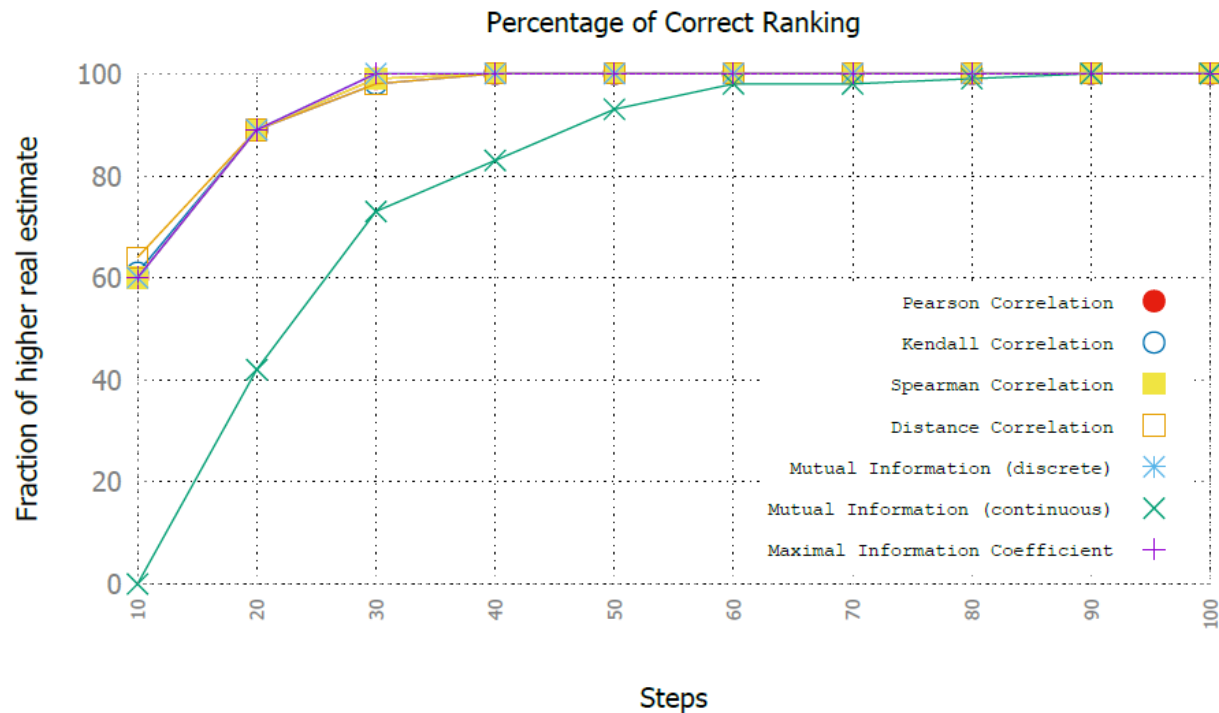
Evaluation of measures for two-man saw scenario

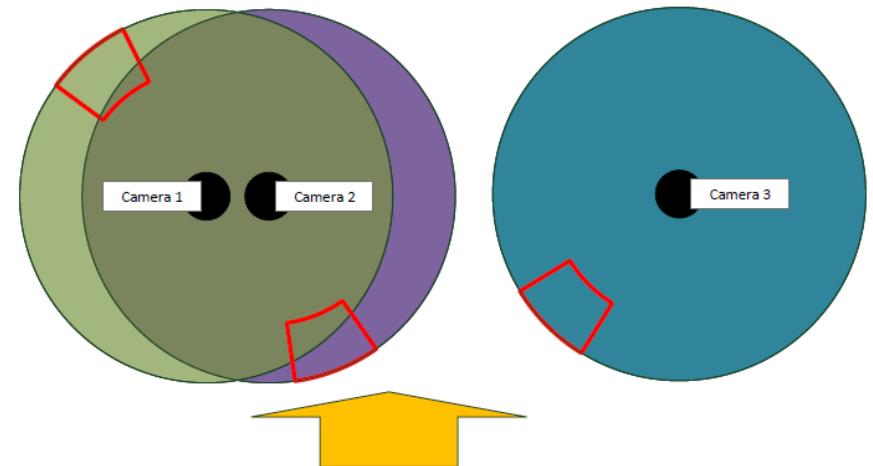
- Detection if a single estimator is used (i.e. all information combined into one estimator) and the own configuration is not considered



Evaluation of measures for two-man saw scenario

- Detection if the own configuration is considered, i.e., there is one estimator for each of the configurations (Push and Pull)

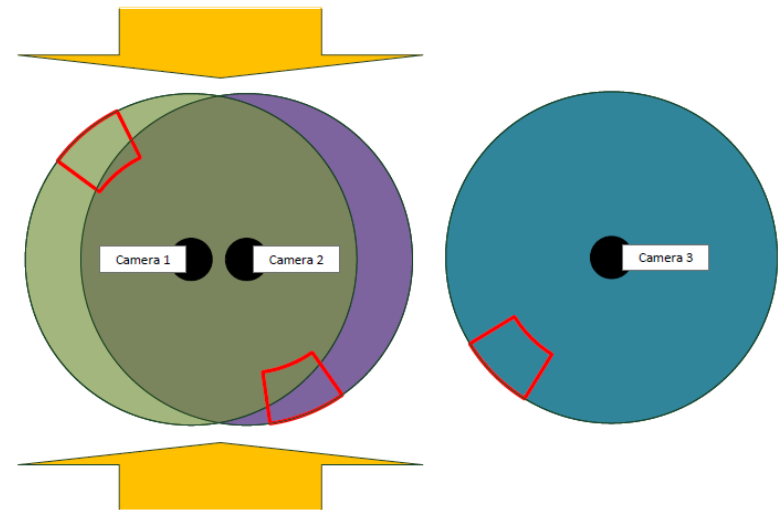




Scenario A

- Top-down view on a smart camera network
- Black dots depict cameras surrounded by a circle
- Circle marks potentially observable area of a camera
- Red shapes show the field of view for an exemplary PTZ configuration
- Yellow arrow indicates from where and in which direction the objects of interest move

Here: the flow of targets represented by the yellow arrows is changed; the challenge is more complex.



Scenario B

- Top-down view on a smart camera network
- Black dots depict cameras surrounded by a circle
- Circle marks potentially observable area of a camera
- Red shapes show the field of view for an exemplary PTZ configuration
- Yellow arrow indicates from where and in which direction the objects of interest move

Which parameter has the highest impact for ...

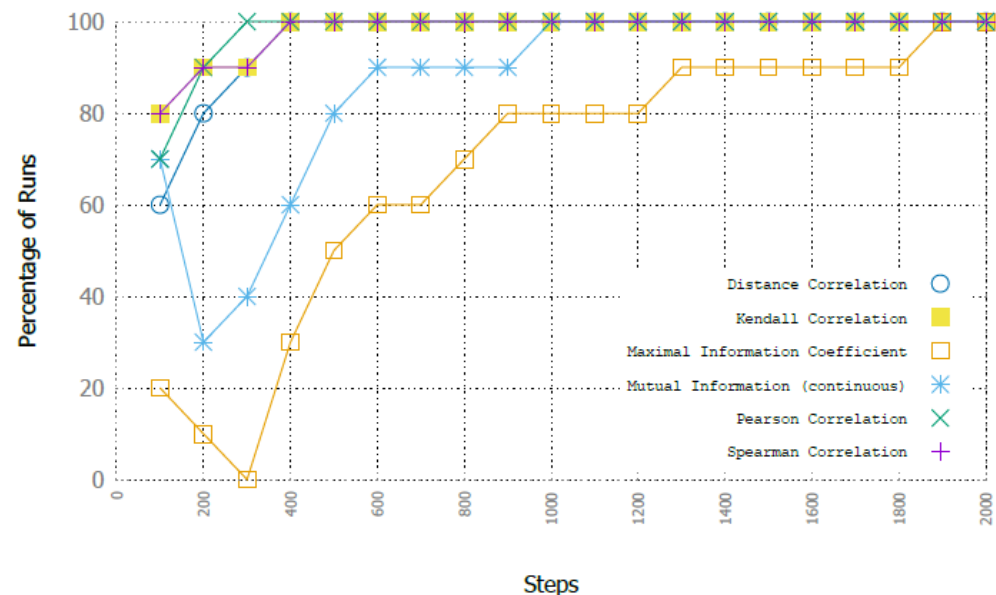
- coverage optimisation?
- object tracking?
- 3D reconstructions?

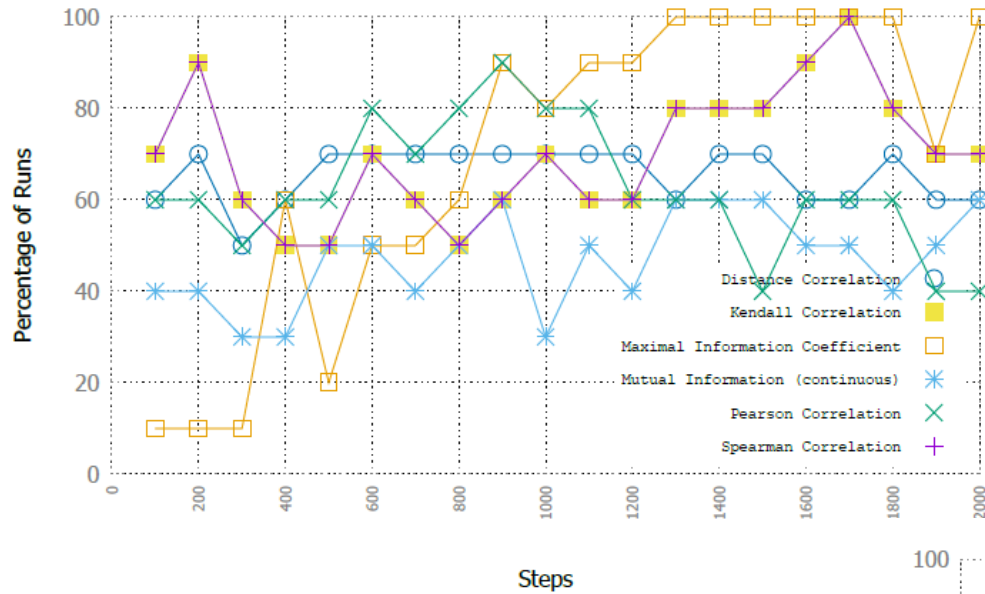
Scenario A

- Pan angles: [0; 360] degrees
Tilt angle: [120; 180] degrees
Zoom: [12; 18]
- Each runs compares influences:
Of Cam2 on Cam1
Of Cam3 on Cam1

The graph shows the fraction of runs in which influence of configuration component (pan, tilt, or zoom) of Cam1 on Cam0 is detected to be higher than the influence of Cam2

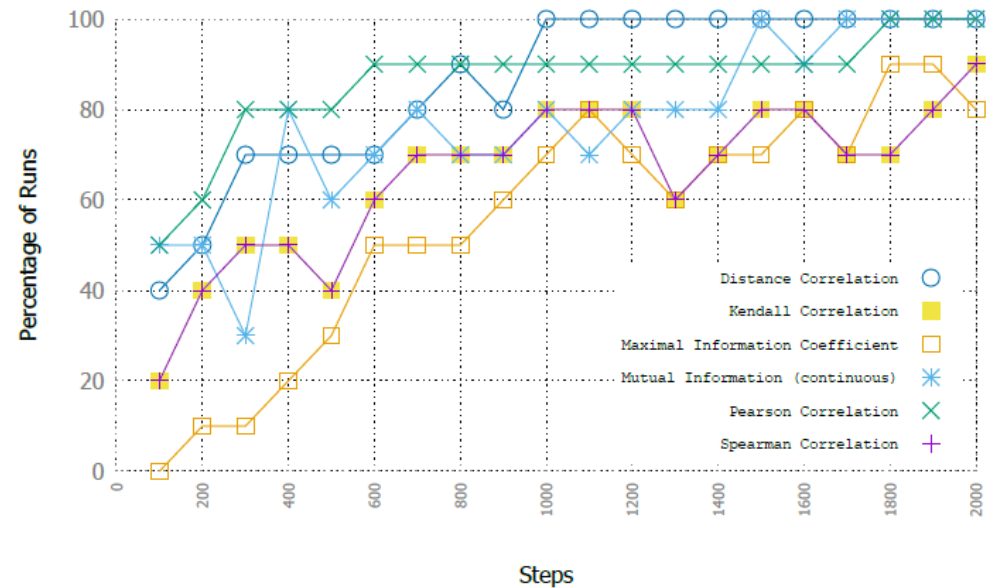
Impact of the Pan





Impact of the Zoom

Impact of the Tilt

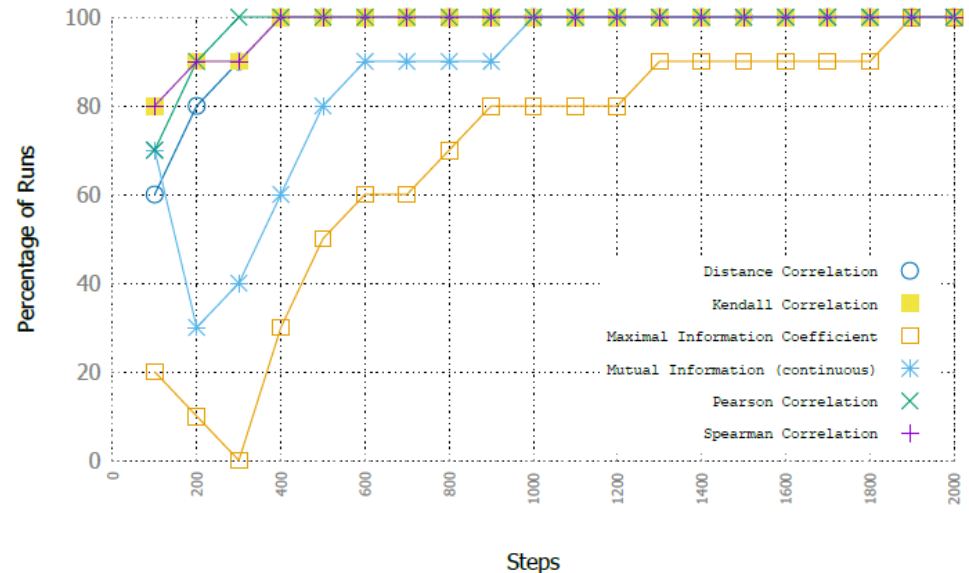


Observations:

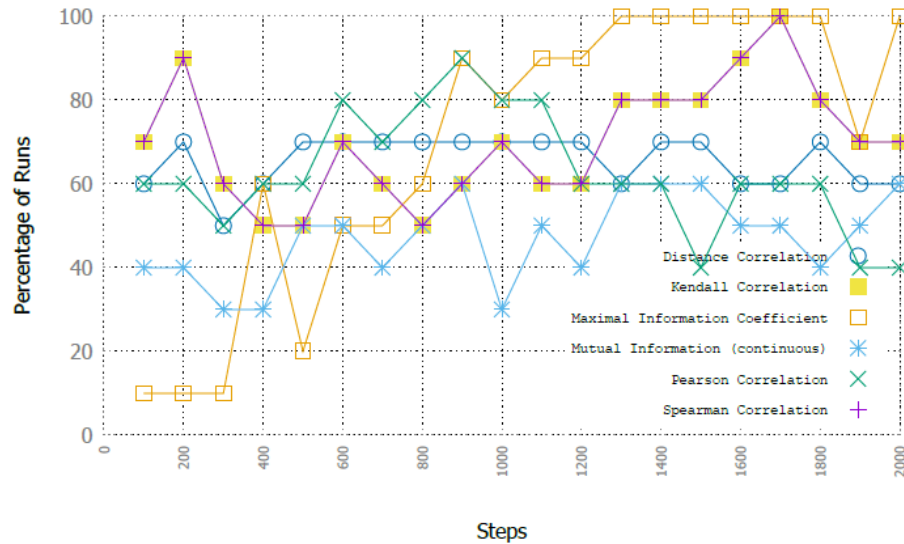
- After a few 100 steps, most measures allow a correct detection in 100% of the cases
- Continuous mutual information and maximal information coefficient take a little longer to reach this level of certainty
- Pan: We see that the distance correlation shows the best result compared to the other measures.
- However, the other measures work as well with higher sample size.
- Even though the MIC finds an influence quite reliably, the other measures do not show a definite result which can be explained by the minor influence of the configuration components, i.e., in most cases, the **zoom does not determine if there is a positive reward** at all but only the absolute value of this positive reward.

Scenario B

- Graph shows the fraction of runs in which influence of configuration component (pan, tilt, or zoom) of Cam1 on Cam0 is detected to be higher than the influence of Cam2

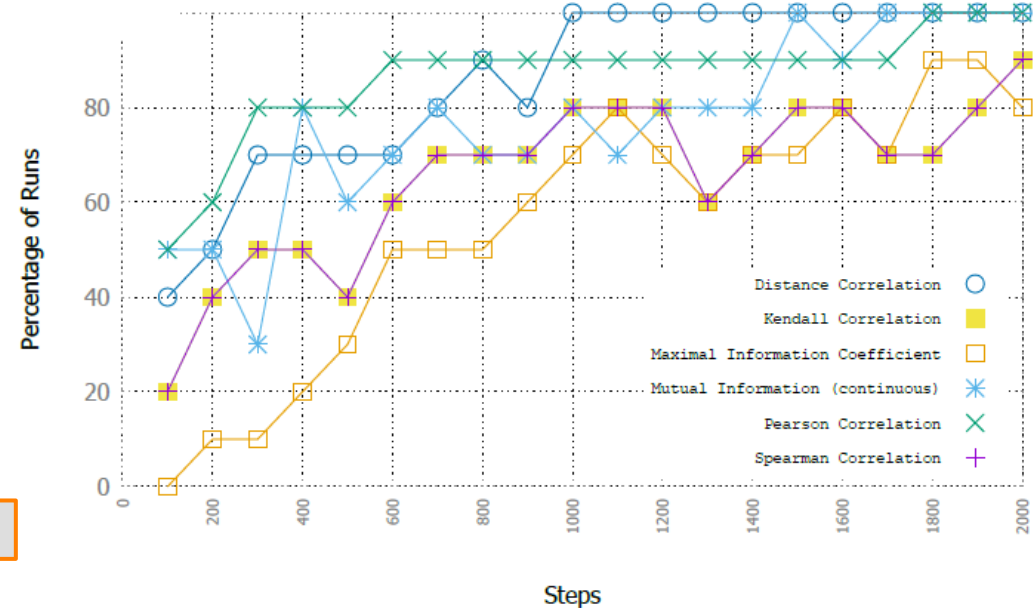


Impact of the Pan

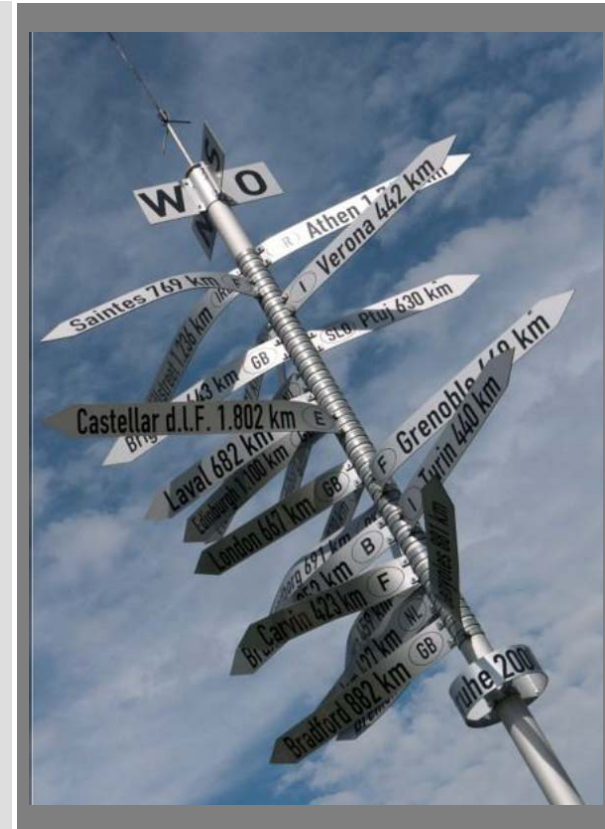


Impact of the Zoom

Impact of the Tilt



- Motivation
- Running example and approach
- Detecting influences and dependencies
- Behaviour of measures in test cases
- **Influence detection at runtime**
- Influence-aware self-adaptation
- Conclusion and further readings



Motivation

- Obviously, it is hardly possible to foresee all situations a system will face at runtime.
- Especially if the considered system has to act in parallel to other systems (that may be under the control of another stakeholder) and both should be able to adapt to new goals or environmental changes.
- Means: We need to focus on the detection of influences at runtime.
- One challenge: The problem of static behaviour of the systems that introduce disturbances in the measurement by causing correlations that are not based on causality but coincidence.

Runtime influence detection

- Influence detection algorithm **relies on dependency measures** that estimate the **correlation** between two random variables
- Applying this directly at runtime may lead to wrongly detected influences because such **correlations can appear without an underlying causality**
- Previous experiments:
 - It is possible to infer causality from the correlation between the actions of two subsystems
 - Reason: We enforced the configurations to be randomly selected from independent uniform distributions
- However: For runtime detection, it has to be considered that we can face **autocorrelations and other disturbances** within the configurations of the systems

Approach:

- Rely mostly on randomised configurations that appear naturally in applications that use **reinforcement learning** during runtime
- Reason:
 - Need for **exploration** in these tasks to find optimal strategies
 - Avoid to "get stuck" in a local optimum
 - Happens often with **greedy approaches** because the algorithm only tries one behaviour and if it is "good" it sticks to it and misses out on other strategies that may lead to better results
- Easy and reliable approach: Use **ϵ -greedy action selection**
 - Stick to the action that it has evaluated as the best one so far most of the time
 - With the probability of ϵ it will use a random action regardless of the so far evaluated usefulness of it

Approach

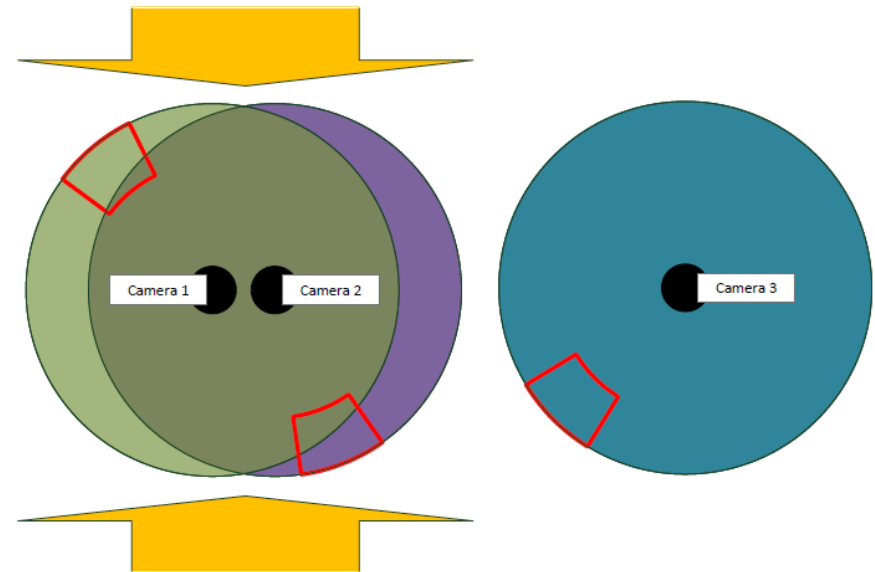
- This means a natural approach to the issue of correlation without causality is to only use the samples that are formed from such exploration steps.
- But this would lead to a significantly **lower amount of samples**.
- To avoid this, we also have to examine **how much “randomness”** is necessary to allow the measurement to function properly by conducting experiments with different levels of ϵ .
- Identifying the **lowest level of ϵ** will ensure the fastest detection of influences without running the risk of falsifying the measurement.

Using ϵ -greedy strategies

- One might expect: A very high level is crucial for a successful detection
- But: **Correlation mainly appears due to the repetition of specific patterns** in both systems in the same frequency
- If such **patterns** are **brought out of sync**, a correct detection is possible despite the repetition, i.e., a high autocorrelation does not necessarily mean that the measurement is flawed
- Example:
 - Two systems A and B
 - Each switch back and forth between their two configurations 1 and 2
 - System A gets a reward of 0.5 if it applies 1 and 0.8 if it applies 2
 - If both systems switch their configurations at the same time step, the measurement will be falsified because system B's configuration will correlate with the reward of system A
 - If one of the systems randomly does not switch the configuration the behaviour becomes "asynchronous" and the measurement works well

Experimental analysis

- Three cameras, where Cam1 and Cam2 influence each other due to an overlap in their potential observable space
- Cam3 does not overlap with one of the other cameras: no influence on the other two
- Previously: Detection works when the configuration is chosen randomly from the full configuration space



Experimental analysis

- Each SC is controlled by a Q-Learning algorithm

- Reminder:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

- s_t : **State** of time t
- a_t : **Action** at time t
- $Q(s_t, a_t)$ denotes the old value of the **quality function** (i.e. the estimated value of applying a_t in s_t)
- $t+1$: Next **time** step
- $\alpha \in (0,1]$: **Learning rate** that determines how much the current experience (i.e. the current reward) is taken into account for approximating the Q-value
- $\gamma \in (0,1]$: **Discount factor** that determines the fraction of estimated future rewards that is taken into account in the present step
- r_{t+1} : **Reward** at time $t+1$

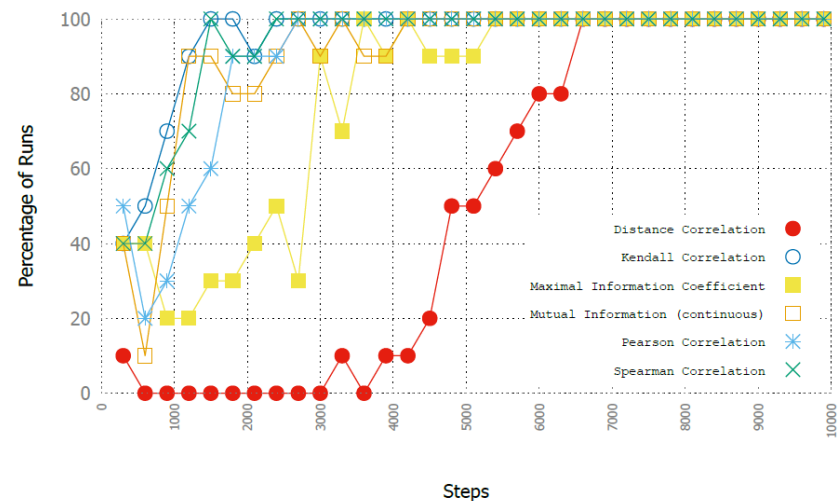
SC controlled by Q-Learning

- Alignment of camera is limited to:
 - 12 **pan** angles $s_p = \{0, 30, 60, \dots, 300, 330\}$
 - 3 **tilt** angles: $s_t = \{120, 150, 180\}$
 - 3 **zoom** levels $s_z = \{12, 18, 24\}$
- Result: $12 \cdot 3 \cdot 3 = 72$ **possible states** per camera
- Actions: Camera can increase, decrease or keep the pan, tilt, and zoom in each time step
- Result: $3 \cdot 3 \cdot 3 = 27$ **possible actions**
- Configuration space is $C = s_p \times s_p \times s_z$



Results: Pan only

- System has run for 10,000 steps
- Applying random actions
- Figure shows the percentage of runs in which Cam1 detects Cam2 as more influencing than Cam3
- Result: Each measure can find the influences very reliably within the 10,000 steps with the distance correlation being much more unreliable using fewer samples



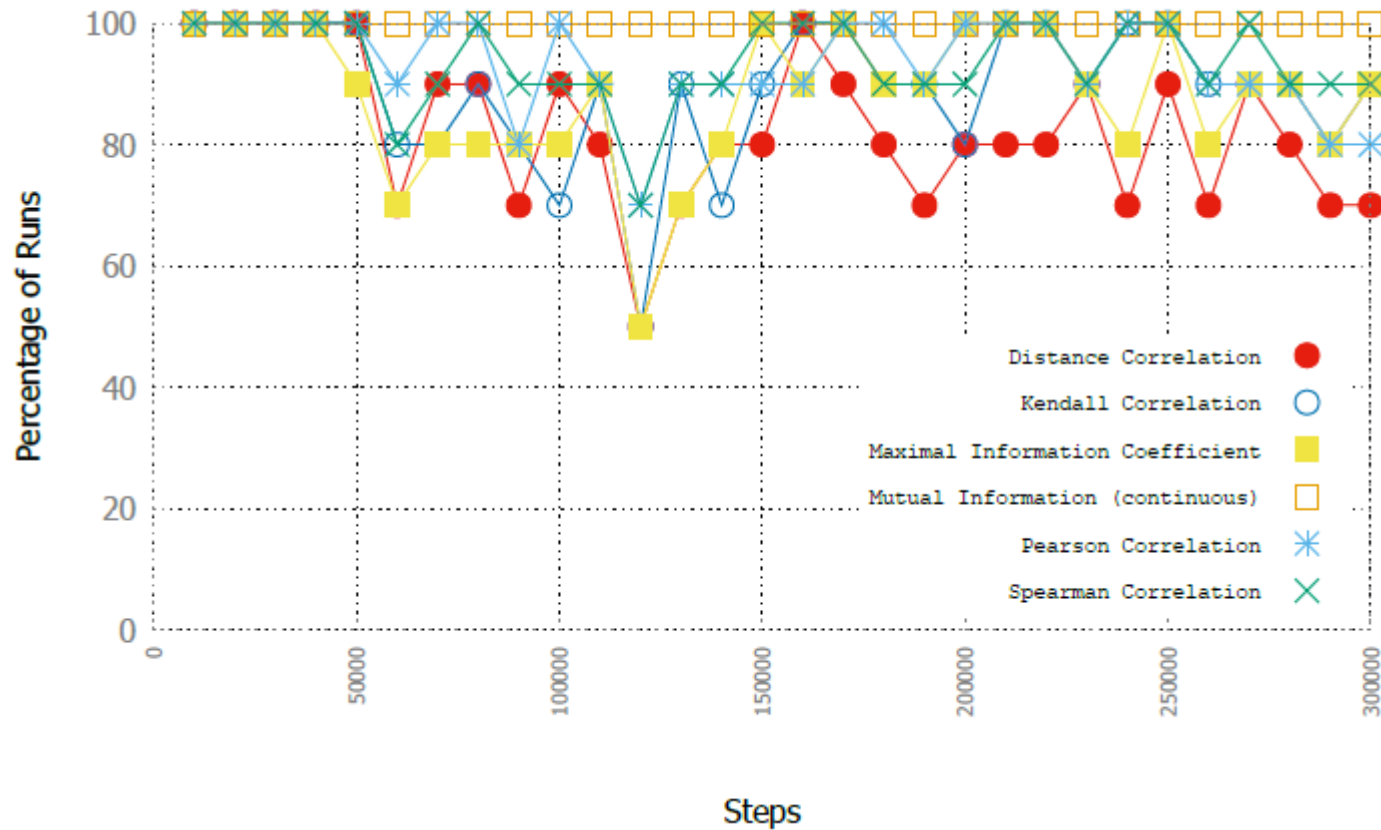
Impact of ϵ

- Analyse how the detection rate changes if we switch from a purely random action selection to ϵ -greedy
- Means: ϵ is varied between 1 and 0.05 as follows:

Step	Start	10k	20k	30k	40k	50k	70k	80k
ϵ	1.0	0.9	0.81	0.73	0.66	0.59	0.53	0.48
Step	90k	100k	110k	120k	130k	140k	150k	160k
ϵ	0.43	0.39	0.35	0.31	0.28	0.25	0.23	0.20
Step	170k	180k	190k	200k	210k	220k	230k	240k
ϵ	0.19	0.17	0.15	0.14	0.12	0.11	0.1	0.09
Step	250k	260k	270k	280k	290k	300k	310k	320k
ϵ	0.08	0.07	0.06	0.06	0.05	0.05	0.05	0.05

- Starts at 1.0 and decreases by 10% every 10k steps until it reaches 5%.

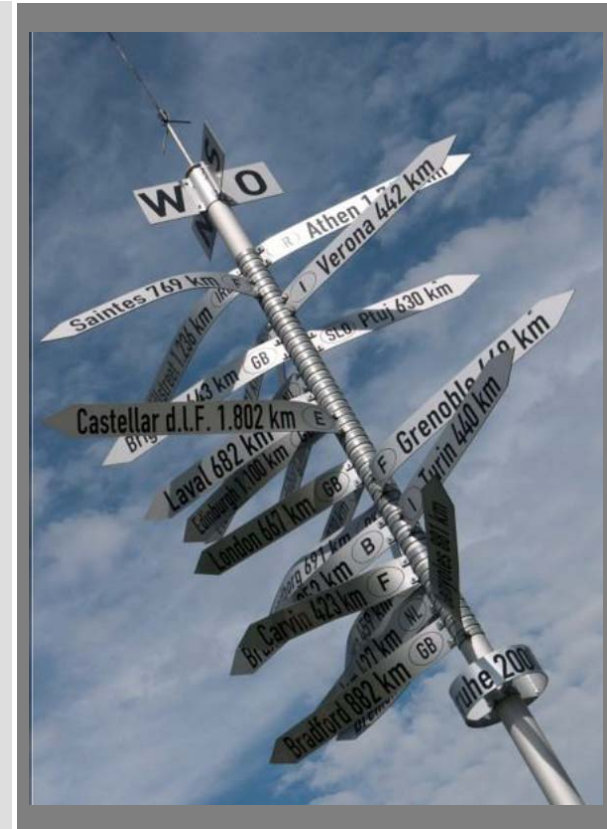
Results



Results

- Detection rate if a single subsystem Q-learning with an ϵ -greedy action selection is used and the last 10, 000 samples are used for the detection
- Q-learning algorithm uses a low $\alpha = 0.1$ to handle fluctuations in the reward signal and a high $\gamma = 0.9$ to allow the learning of a sequence of actions
- Works well for the first 50,000 steps, i.e., for $\epsilon > 0.66$
- Afterwards: Detection works worse for some of the measures with an average of about 80%

- Motivation
- Running example and approach
- Detecting influences and dependencies
- Behaviour of measures in test cases
- Influence detection at runtime
- **Influence-aware self-adaptation**
- Conclusion and further readings



Goal: Act based on discovered mutual influences

- Approach **to self-adaption to these identified influences**, i.e., the exploitation of the influences if they have been discovered
- Here: Focus is not on the presentation of a single algorithm, but on giving a **general methodology based on the RL** model
- But: Example realisation with Q-learning

ALGORITHM 1: The main loop for each subsystem using the influence detection and adaption at runtime.

initialization;

for *each step* **do**

 observe own configuration and reward;

 distribute observations to other subsystems;

if *enough observations* **then**

 estimate influences;

 evaluate estimated influences;

if *influence found* **then**

 adapt learning algorithm;

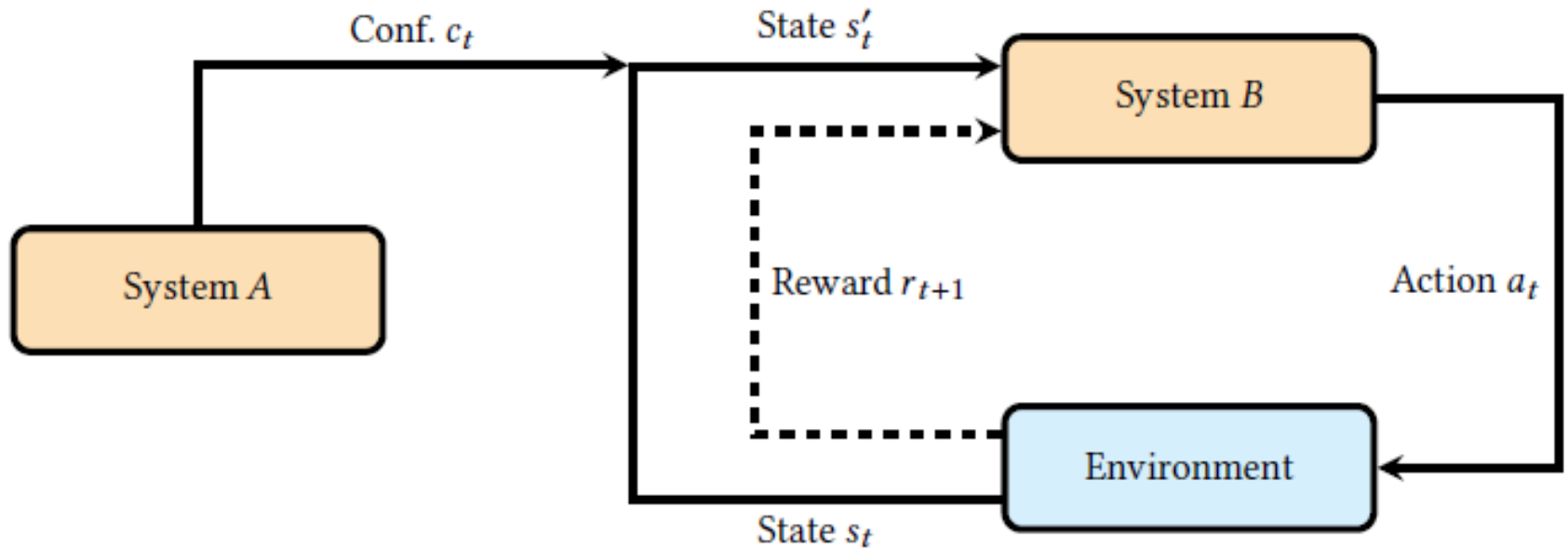
end

end

 learning algorithm decides next action;

 update learning algorithm with new observation;

end

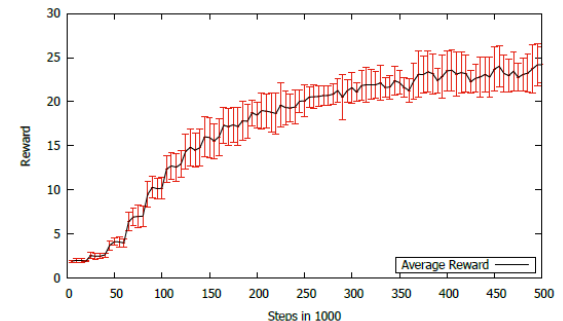
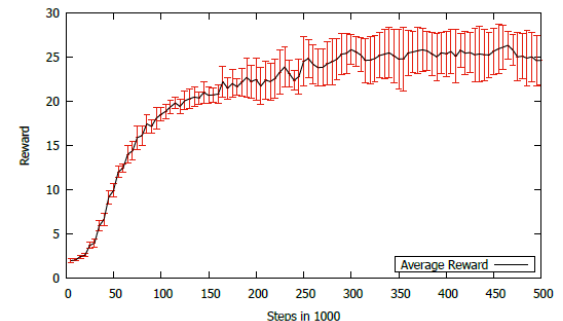
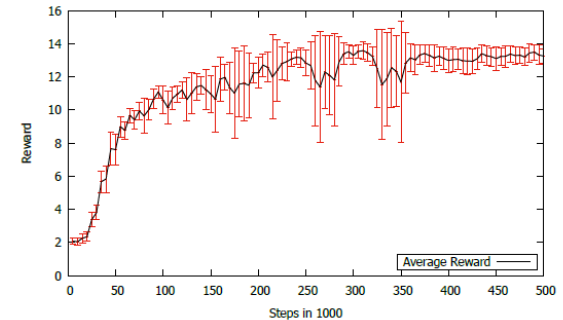


Approach to mutual-influence-aware learning of adaptation behaviour

- Integration of the value of the configuration c_t at time t into the state s_t resulting in the state s_t'
- Example:
 - If B's states are given through a position represented as (x, y) -coordinate between 0 and 1 each, its state is $S = [0,1] \times [0,1]$
 - Assuming the influence detection has found A's configuration component c_B , which represents the speed of B and is between -1 and 1, as influencing, the state space S will be extended to a new state-space
$$S' = S \times [-1, 1] = [0,1] \times [0,1] \times [-1, 1]$$

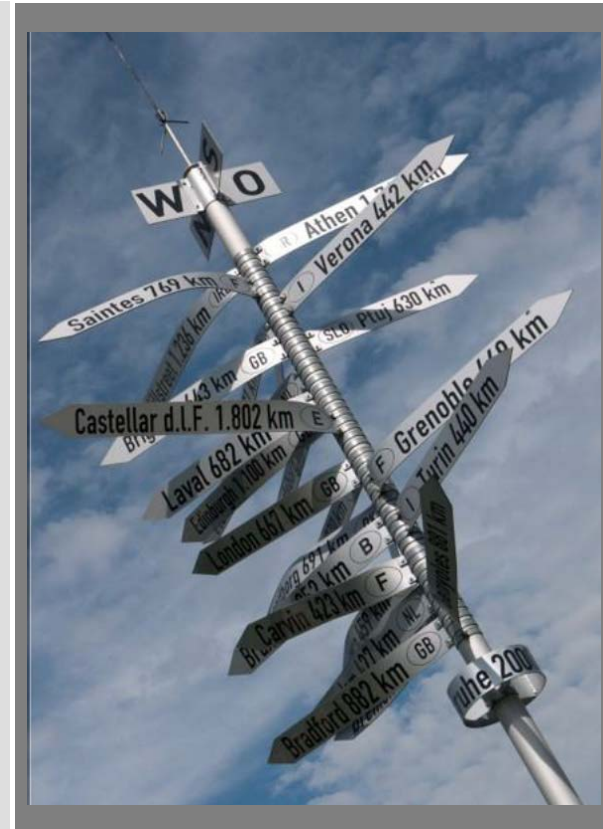
Evaluation:

- Single learners
- State of Cam2 is included in the state of Cam1 from start on
- Configurations are integrated into the state-space of Cam1 dynamically during runtime



What are the major drawbacks of this approach?

- Motivation
- Running example and approach
- Detecting influences and dependencies
- Behaviour of measures in test cases
- Influence detection at runtime
- Influence-aware self-adaptation
- Conclusion and further readings



This chapter:

- Defined mutual influences
- Showed example where they occur
- Explained a basic methodology to detect them based on stochastic dependency measures
- Demonstrated the behaviour of these measures
- Analysed the impact of runtime utilisation
- Provided an example for making use of this information in self-adaptation mechanisms based on reinforcement learning

By now, students should be able to:

- Define and motivate mutual information detection
- Characterise and compare the different dependency measures
- Explain the methodology for mutual influence detection
- Apply detected mutual influence information to self-learning adaptation strategies

Current questions in research related to mutual influences:

- How to deal with time-delayed utility assignments?
- How can the scalability be improved, i.e. a goal-oriented testing of candidates (prioritisation)?
- There are novel (more efficient) correlation measures – can they be applied to the problem? How? What is the impact?
- How can we integrate the mutual information within more sophisticated reinforcement learning techniques such as LCS?
- All current scenarios identify static influences (e.g. fixed camera positions) – what happens if we allow for dynamics (e.g. moving cameras)? How much dynamics is tolerable?
- ...



Just ask, if you're interested in a topic for your thesis or searching for a HiWi position...

Mutual influences:

- Rudolph, Stefan; Tomforde, Sven; and Hähner, Jörg: „On the Detection of Mutual Influences and Their Consideration in Reinforcement Learning Processes“, arXiv electronic publishing, May 2019, available at: <https://arxiv.org/abs/1905.04205>
- Rudolph, Stefan; Tomforde, Sven; Sick, Bernhard; Heck, Henner; Wacker, Arno; and Hähner, Jörg: “An Online Influence Detection Algorithm for Organic Computing Systems”. In: Proceedings of the 28th GI/ITG International Conference on Architecture of Computing Systems – ARCS Workshops, VDE Verlag 2015, pp. 1- 8
- Rudolph, Stefan; Tomforde, Sven; Sick, Bernhard and Hähner, Jörg: “A Mutual Influence Detection Algorithm for Systems with Local Performance Measurement”. In: Proceedings of the 9th IEEE International Conference on Self-adapting and Self-organising Systems (SASO15), held September 2015 in Boston, USA, pp. 144 - 150

Camera example:

- Rudolph, Stefan; Edenhofer, Sarah; Tomforde, Sven and Hähner, Jörg: “Reinforcement Learning for Coverage Optimisation Through PTZ Camera Alignment in Highly Dynamic Environments”. In Proceedings of ICDSC14 – IEEE International Conference on Distributed Smart Camera Systems held in Venice, Italy, November 4-7, 2014, pp. 19:1 – 19:6.

- Any questions...?