Dissertation Defense

Learning to Adapt to Sensor Changes and Failures

Yuan Shi
Department of Computer Science
University of Southern California

Advisor:
Craig Knoblock

Motivation

Unmanned Underwater Vehicle (UUV)

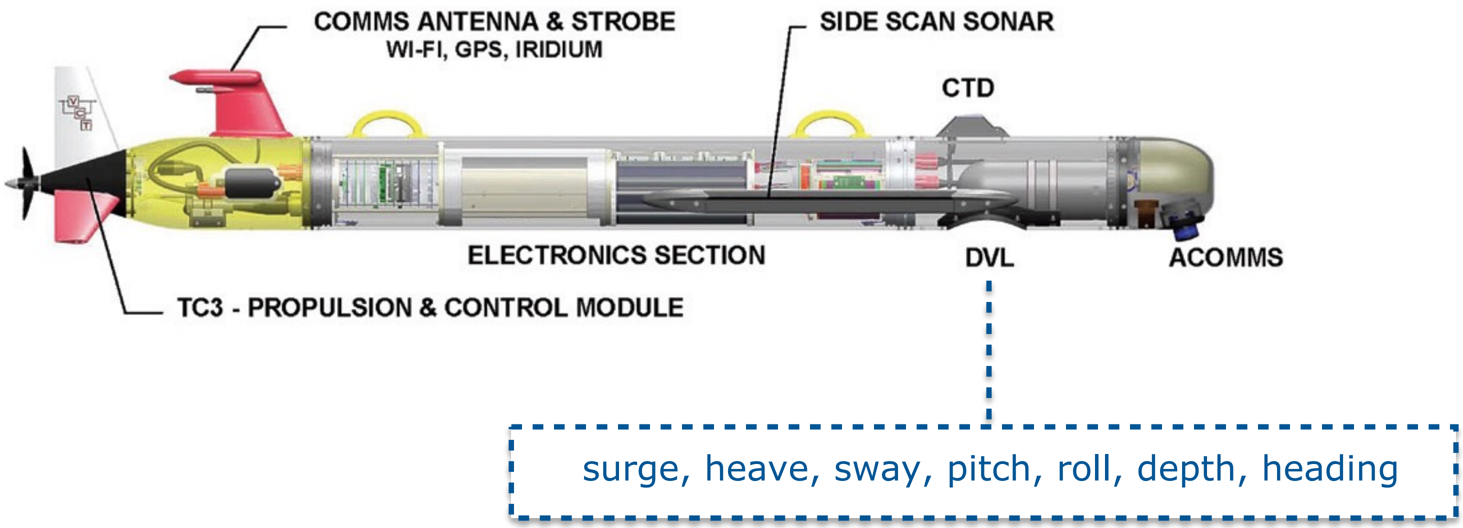
Self-driving Car

Challenges for software systems:
- Changing and uncertain environment
- System failures and changes

How to build **long-lived, survivable** software systems?

Motivation

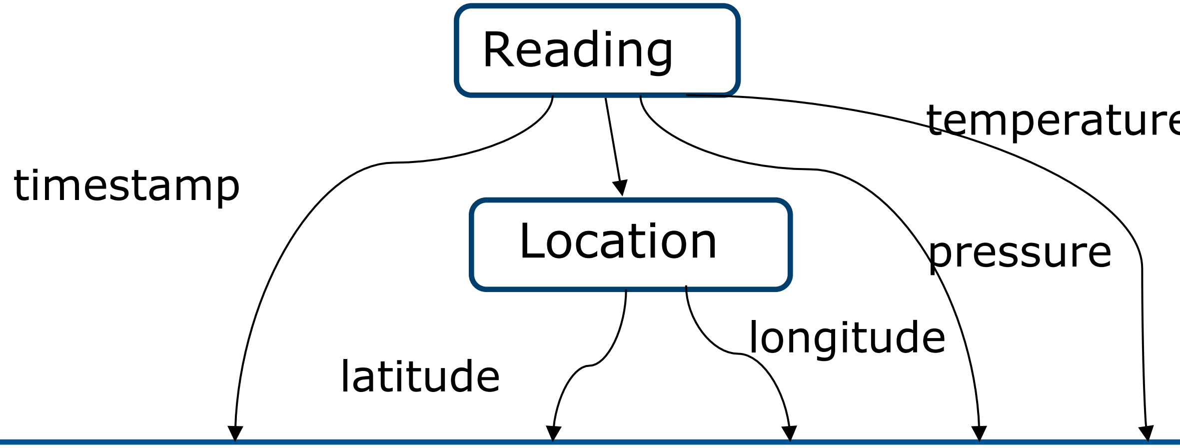
- **Long-lived, survivable software systems**: **automatic adaptation to changes**
  - Significantly reducing maintenance cost
  - Goal of the DARPA BRASS (Building Resource Adaptive Software Systems) program
- Our focus: **adaptation to sensor changes and failures**



Thesis Statement

This thesis proposes a series of machine learning approaches for automatically adapting to sensor failures and changes. These approaches exploit sensor relationships and can address failures/changes in both individual sensors and compound sensors. They empirically achieve high adaptation accuracy on the weather and UUV domains.

Real-world Example of Sensor Failures

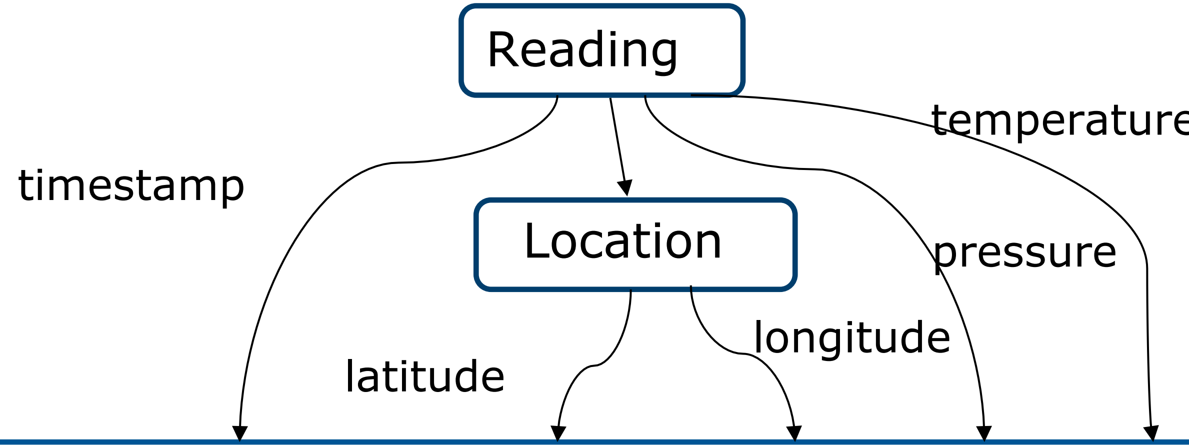


Compound Sensor

2015-04-25 15:07	33.292	118.541	35.2	26.2
2015-04-25 15:12	33.274	118.532	34.8	10.1

Real-world Example of Sensor Failures

Compound Sensor

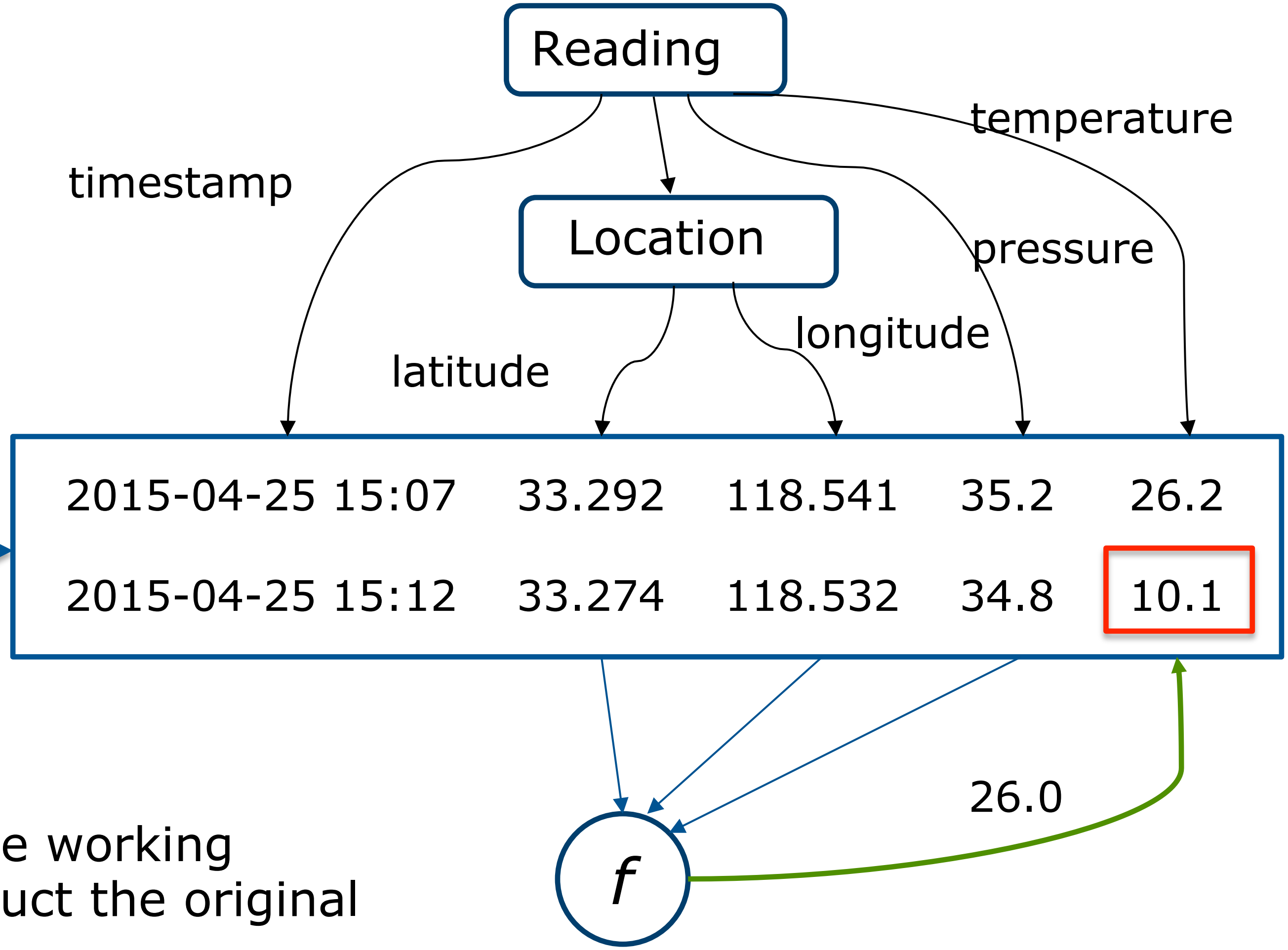


2015-04-25 15:07	33.292	118.541	35.2	26.2
2015-04-25 15:12	33.274	118.532	34.8	10.1

Sensor Failure

Real-world Example of Sensor Failures

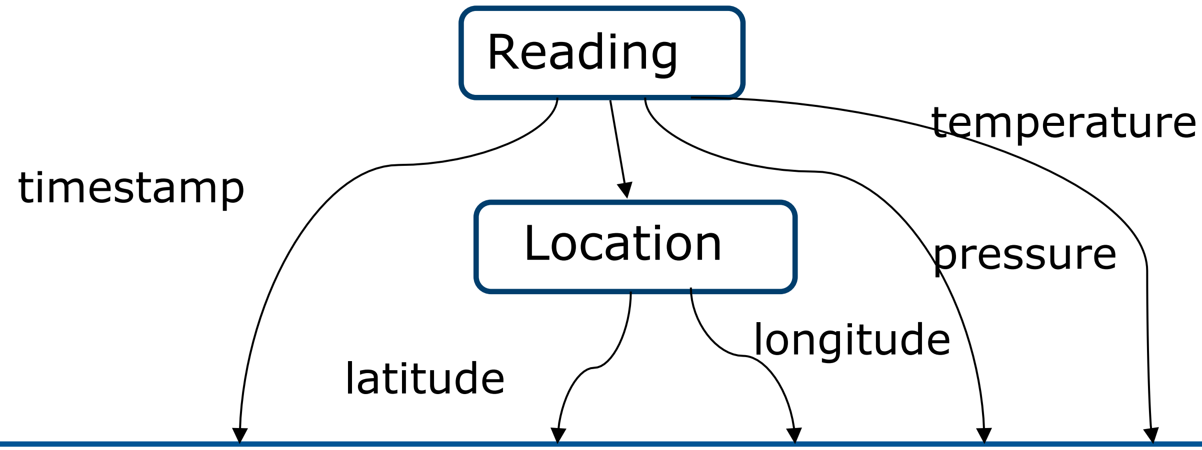
Compound Sensor



Can we combine the working sensors to reconstruct the original temperature?

Sensor Failure

Real-world Example of Sensor Changes



Compound Sensor

2015-04-25 15:07	33.292	118.541	35.2	26.2
2015-04-25 15:12	33.274	118.532	34.8	26.9

Replaced by a new temperature sensor

Sensor Change

Real-world Example of Sensor Changes

Reading

timestamp

Location

temperature

pressure

latitude

longitude

Compound Sensor

2015-04-25 15:07	33.292	118.541	35.2	26.2
2015-04-25 15:12	33.274	118.532	34.8	26.9

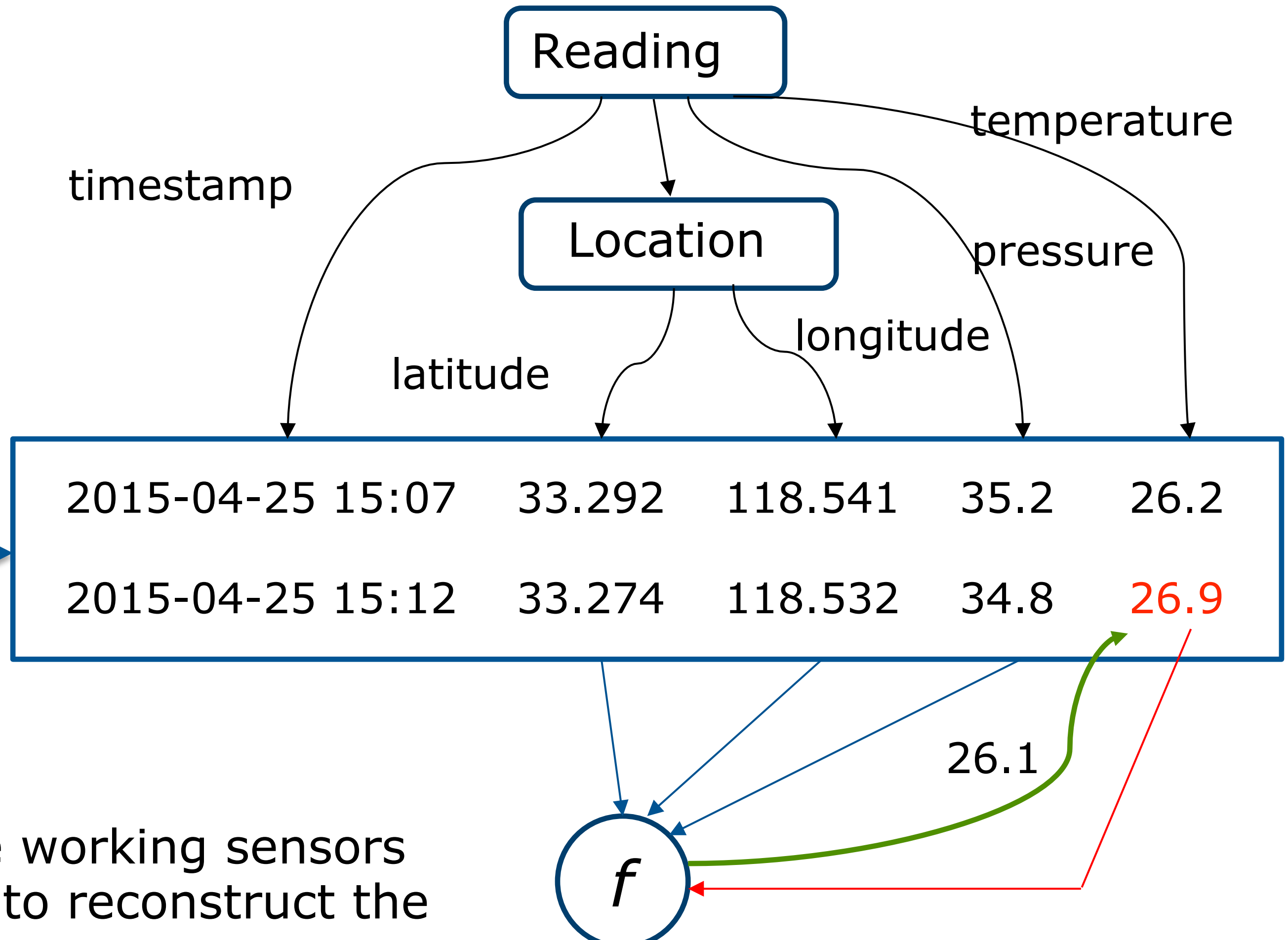
Replaced by a new temperature sensor

Direct replacement can be bad!

Sensor Change

Real-world Example of Sensor Changes

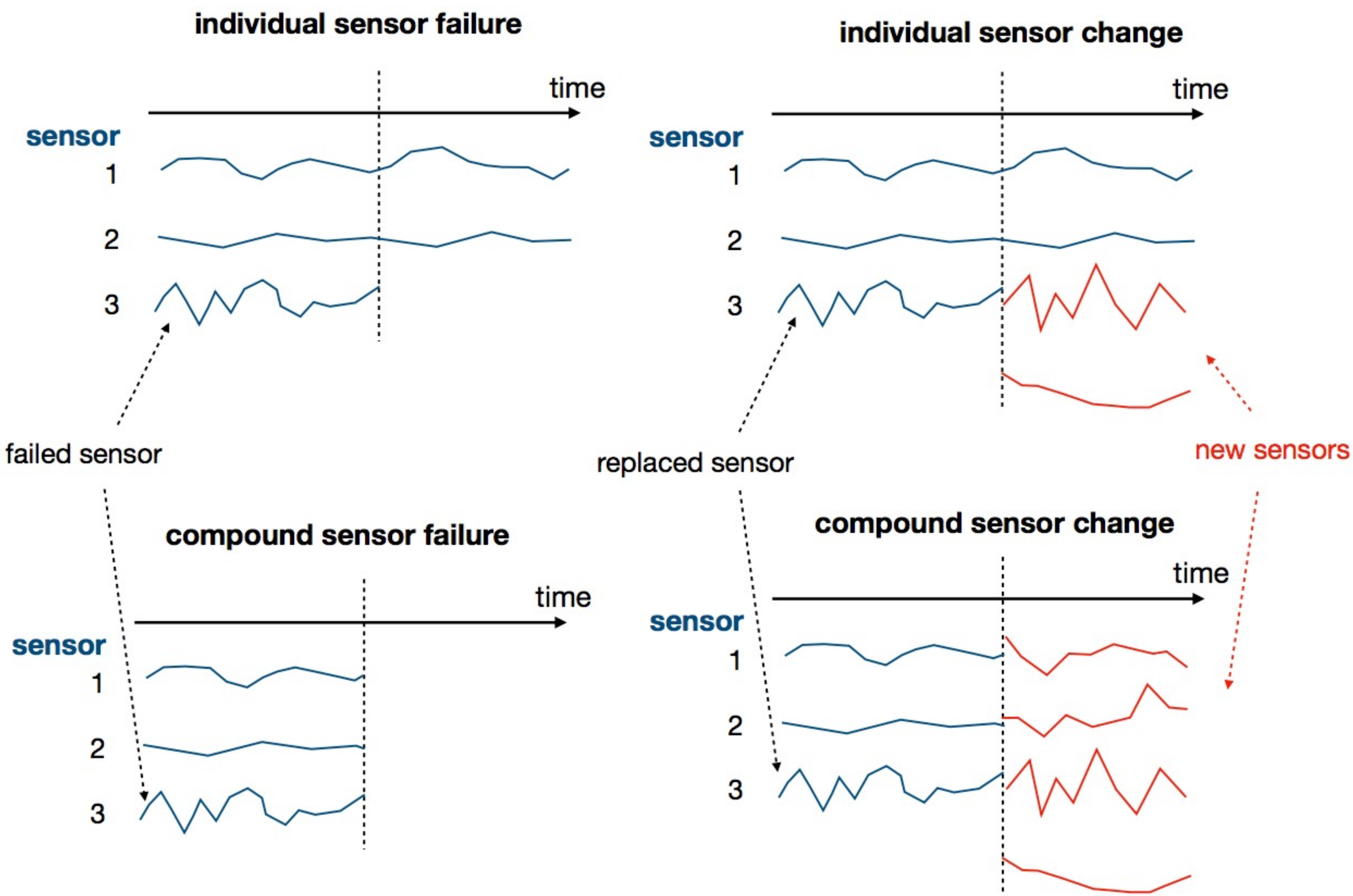
Compound Sensor



Can we combine the working sensors and the new sensor to reconstruct the original temperature?

Sensor Change

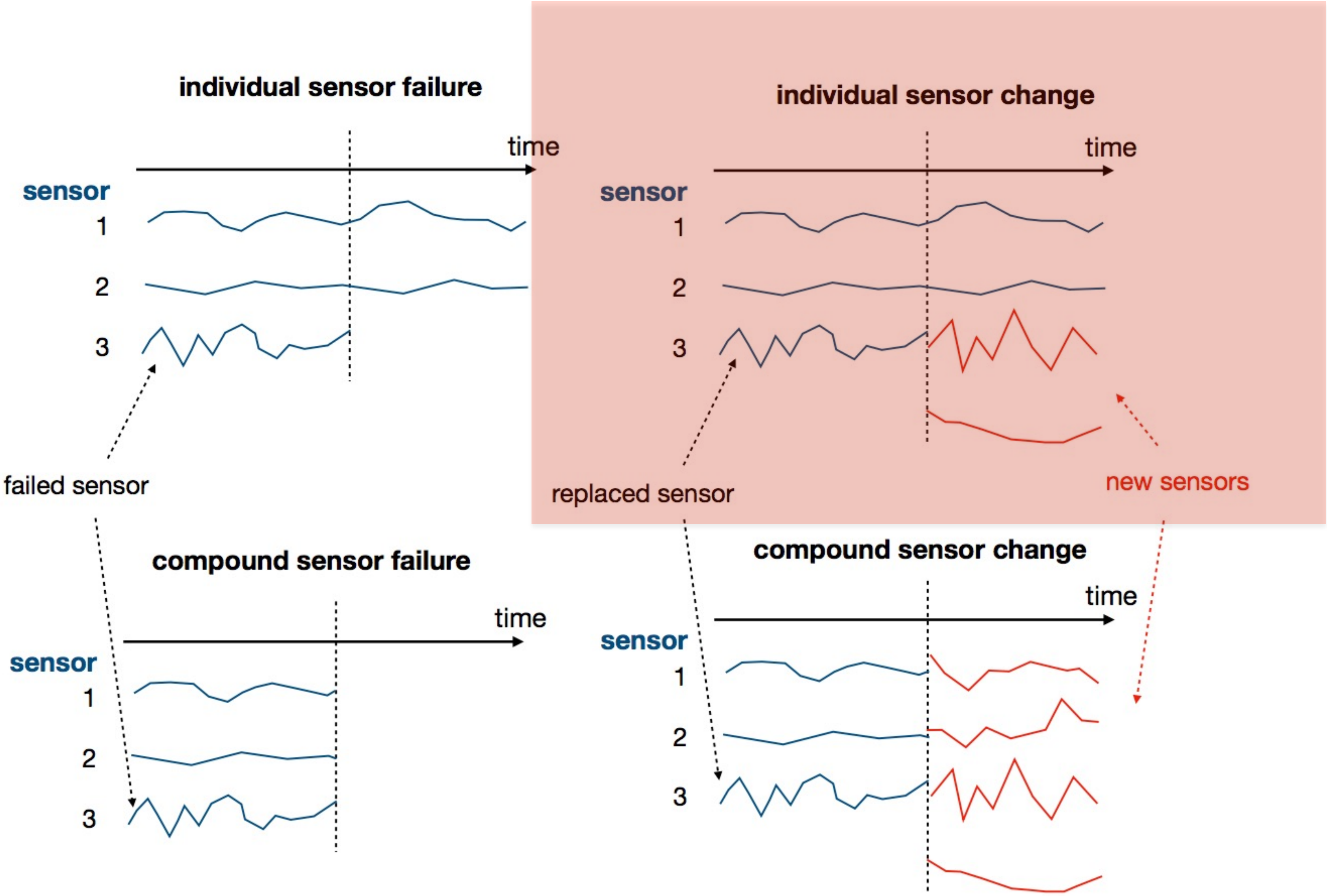
Scenarios of Sensor Failures and Changes



Sensor-level and Model-level Adaptation

- **Sensor-level Adaptation**: reconstructing original sensor values
  - No change for upper-level software
- **Model-level Adaptation**: directly adapting software components (e.g. a classifier) that are built on sensor values
  - Domain adaptation (adapting a model from a source domain to a different target domain) [Daume III and Marcu, 2006] [Pan and Yang, 2010]
  - May be feasible when sensor-level adaptation is not

Scenarios of Sensor Failures and Changes



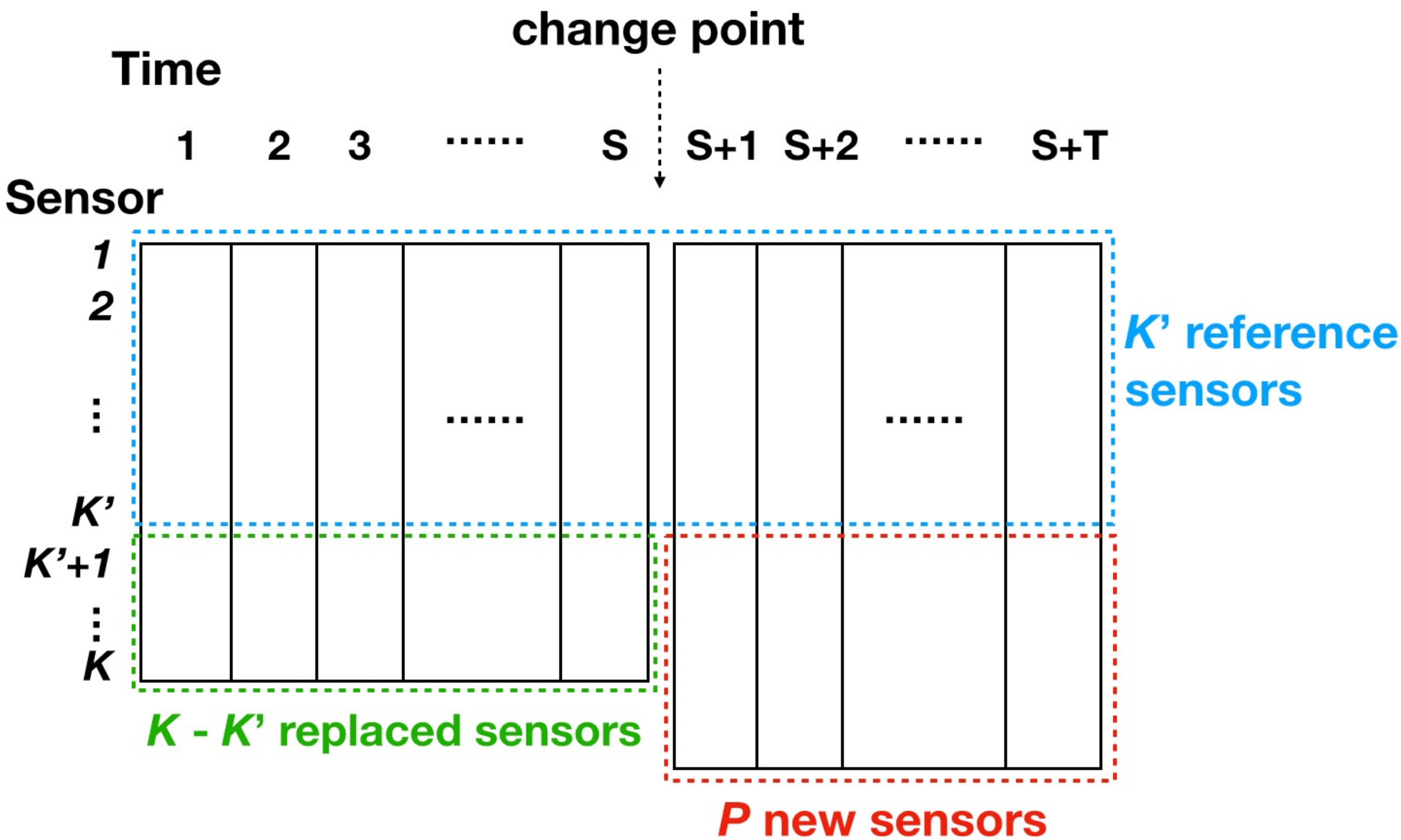
Outline

- Sensor-level Adaptation to Sensor Changes
  - **Yuan Shi**, T. K. Satish Kumar and Craig Knoblock. Automatic Adaptation to Sensor Replacements. FLAIRS-32, 2019
  - **Yuan Shi** and Craig Knoblock. Learning with Previously Unseen Features. IJCAI, 2017
- Model-level Adaptation to Sensor Changes
  - **Yuan Shi** and Fei Sha. Information-Theoretical Learning of Discriminative Clusters for Unsupervised Domain Adaptation. ICML, 2012
- Joint Detection and Adaptation to Sensor Failures
  - Avi Pfeffer , Curt Wu , Gerald Fry , Kenneth Lu, Stephen Marotta, Mike Reposa, **Yuan Shi**, T. K. Satish Kumar, Craig Knoblock, David Parker, Irfan Muhammad and Chris Novakovic. Software Adaptation for an Unmanned Undersea Vehicle. IEEE Software, 2019
  - **Yuan Shi**, T. K. Satish Kumar and Craig Knoblock. Constraint-Based Learning for Sensor Failure Detection and Adaptation. ICTAI, 2018

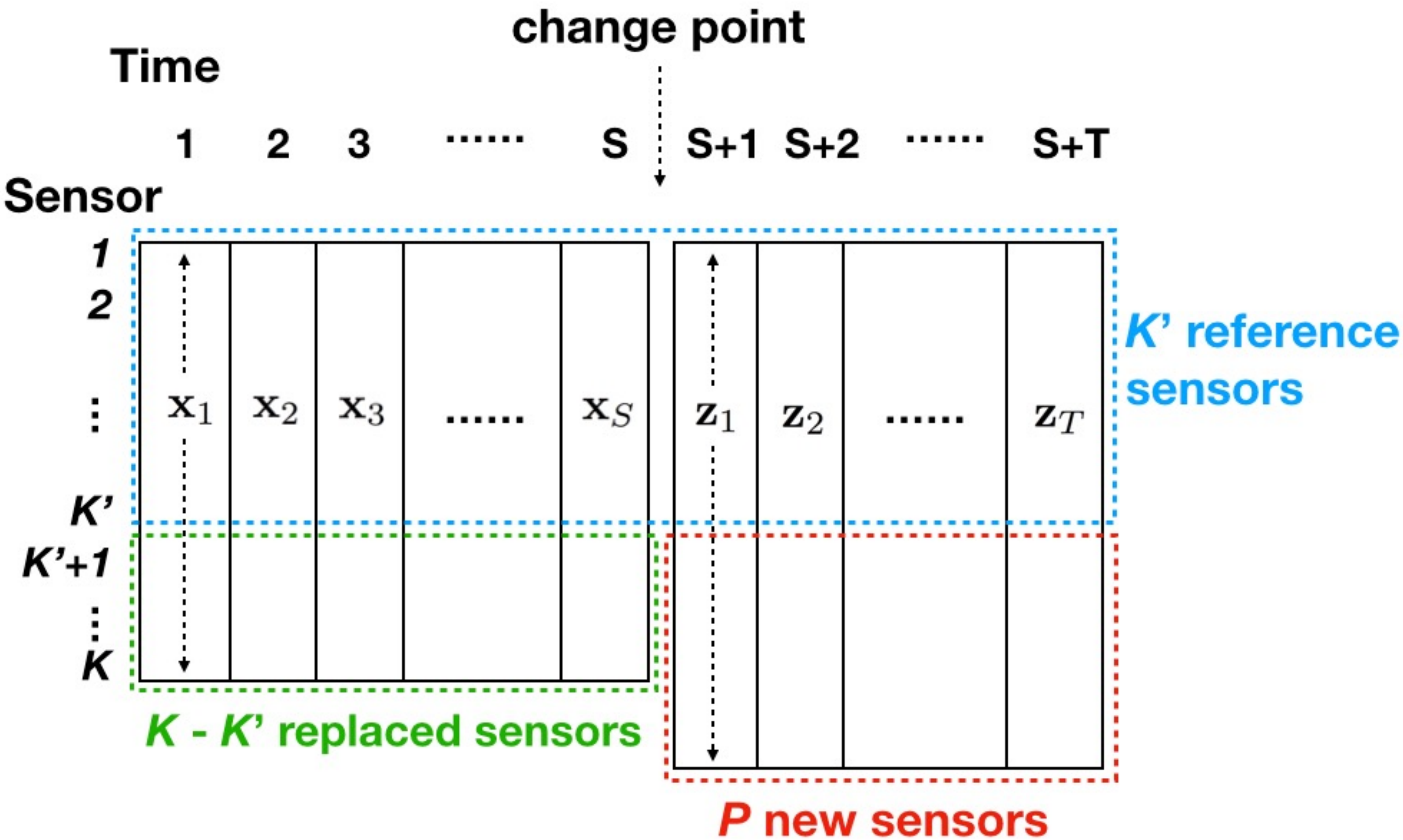
Outline

- **Sensor-level Adaptation to Sensor Changes**
- Model-level Adaptation to Sensor Changes
- Joint Detection and Adaptation to Sensor Failures

Notations of Individual Sensor Changes

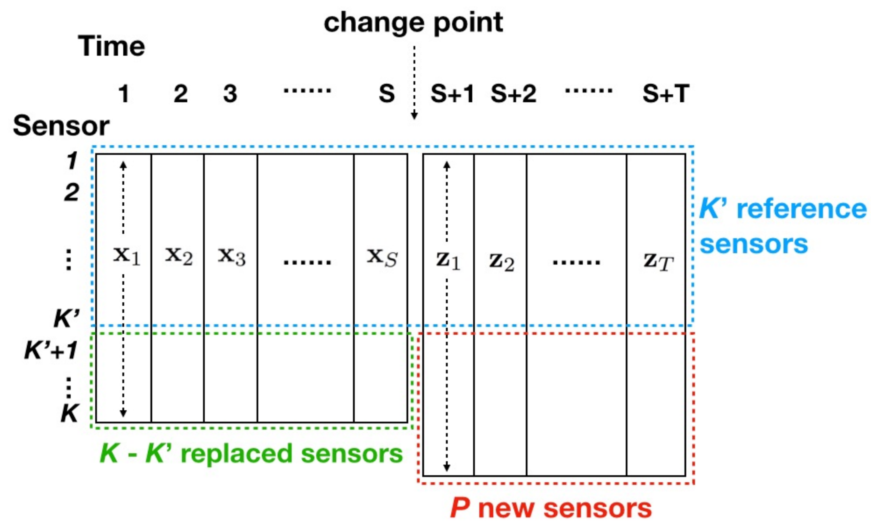


Notations of Individual Sensor Changes



Sensor-level Adaptation to Individual Sensor Changes

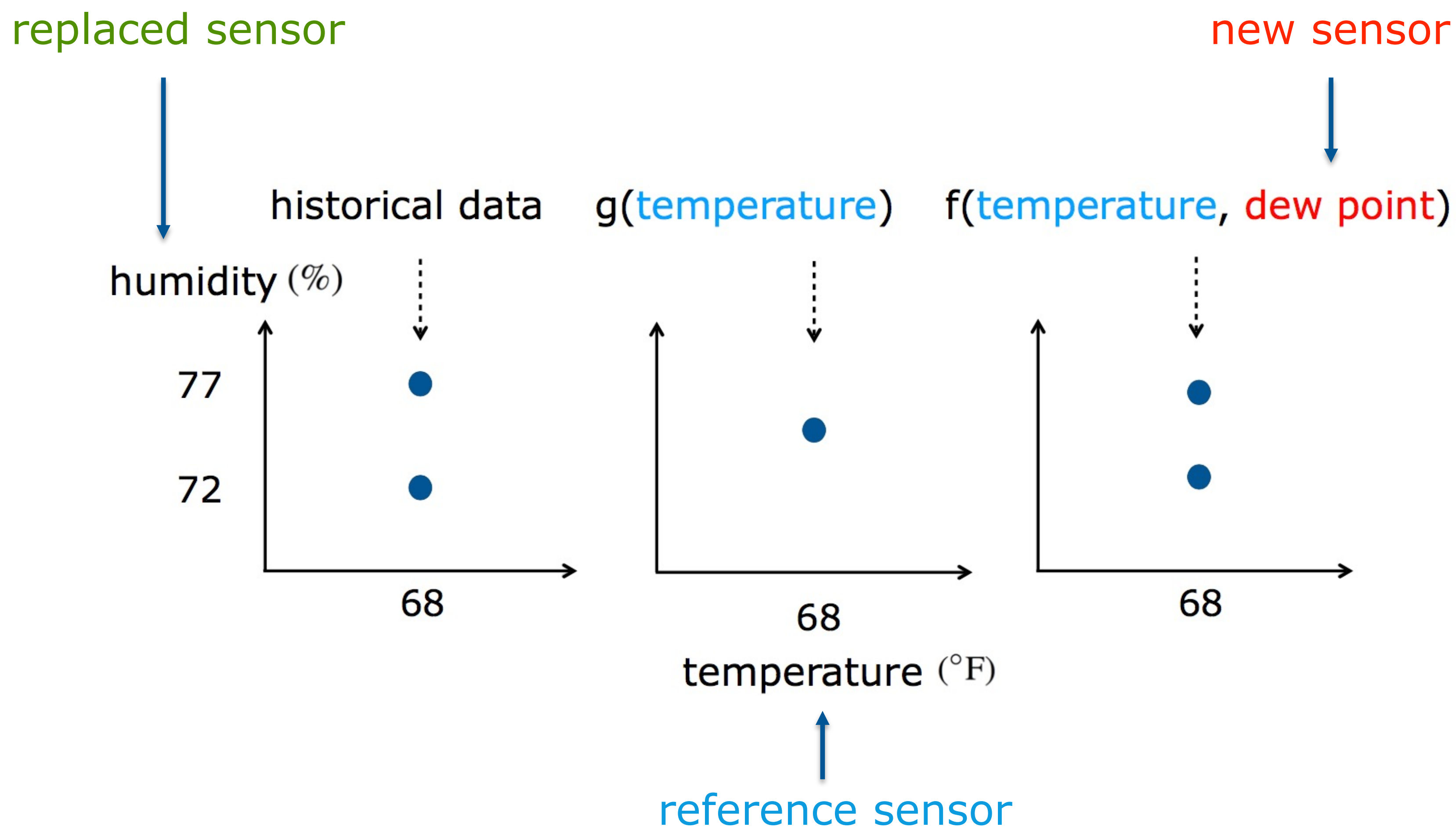
Unexplored in previous work



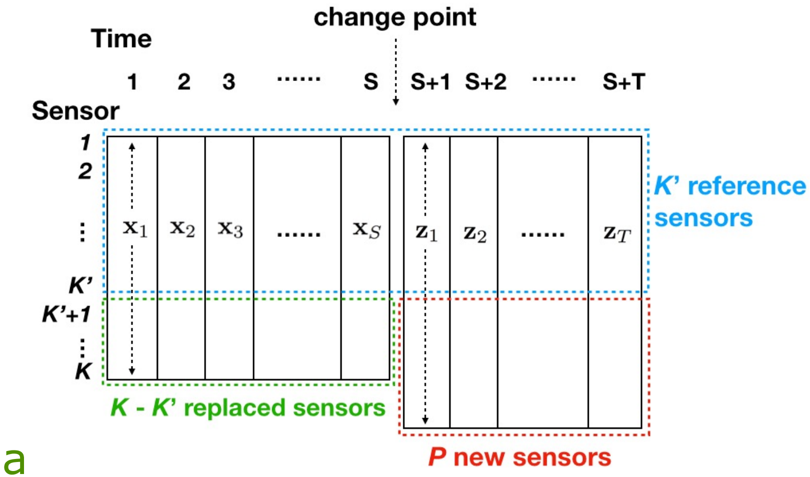
Goal: learning a **reconstruction function**:

f(reference sensors, new sensors) → replaced sensors

Intuition of Exploiting New Sensors



Sensor-level Adaptation to Individual Sensor Changes

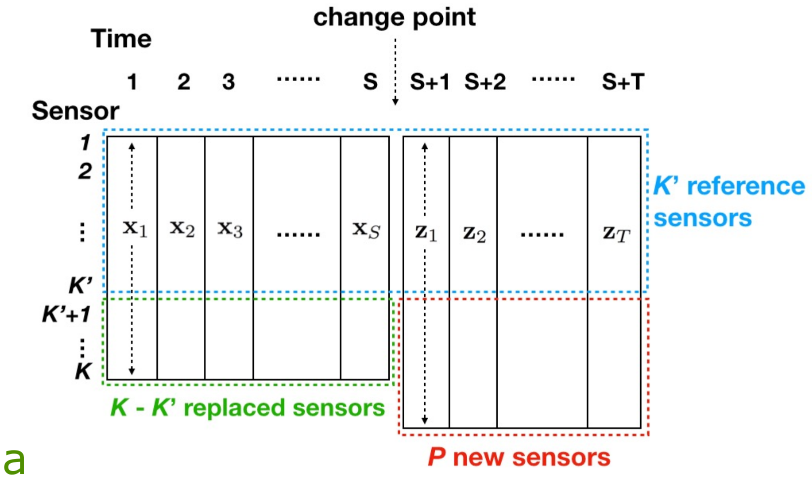


Unexplored in previous work

Reconstruction function:
f(reference sensors, new sensors) repla

Challenge: **no overlapping** between the replaced sensors and new sensors

Sensor-level Adaptation to Individual Sensor Changes



Unexplored in previous work

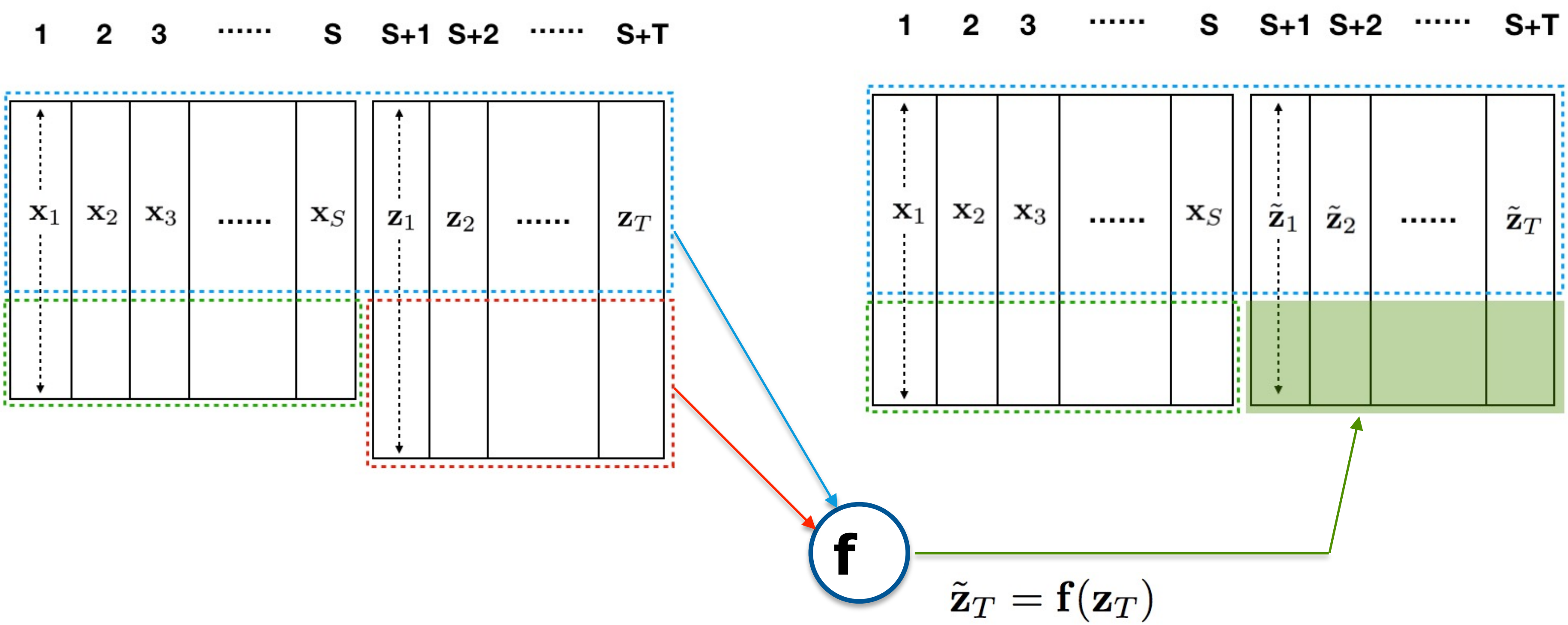
Reconstruction function:
f(reference sensors, new sensors) $\longrightarrow$ repla

Challenge: **no overlapping** between the replaced sensors and new sensors

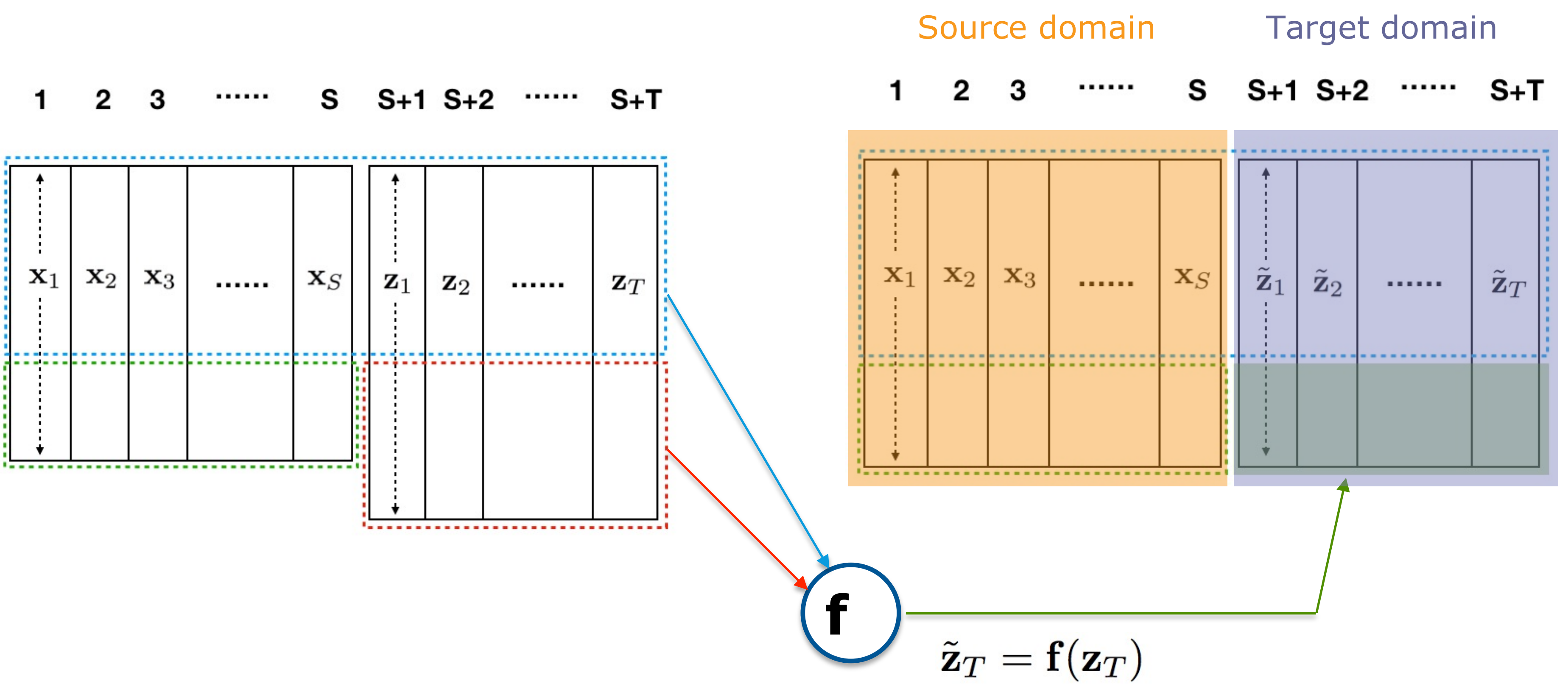
Idea: using the reference sensors as a bridge

Assumption:
1. Sensor values from reference sensors are correlated with those from replaced sensors
2. Sensor values from reference sensors are correlated with those from new sensors

Methodology of ASC (Adaptation to Sensor Changes)

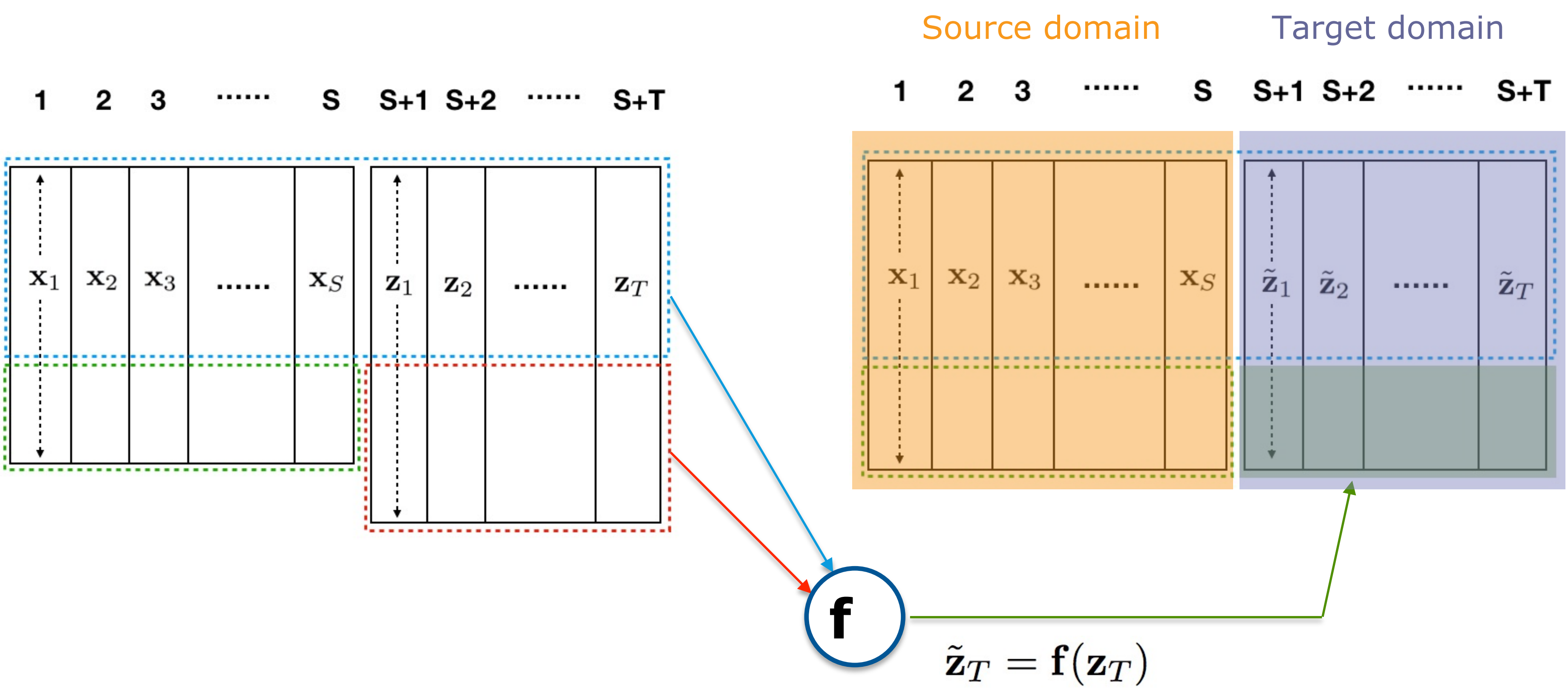


Methodology of ASC (Adaptation to Sensor Changes)



Methodology of ASC (Adaptation to Sensor Changes)

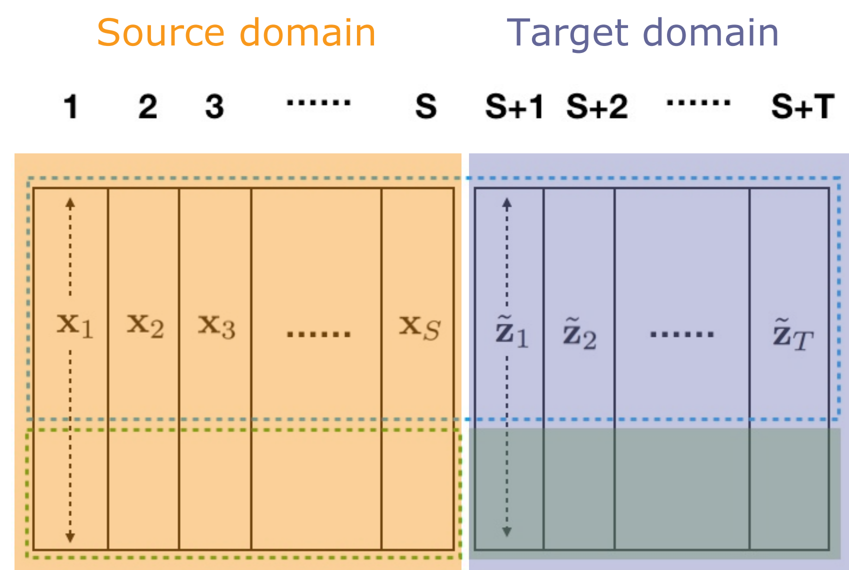
Samples in the two domains distribute similarly



Methodology of ASC (Adaptation to Sensor Changes)

Samples in the two domains distribute similarly

Two sets of samples mixed as much as possible



Minimize cross-domain *k*-nearest neighbor distances

$$\min_{\theta} \sum_{s} \sum_{t \in \mathcal{N}_T^k(s)} \mathcal{D}(\mathbf{x}_s, \tilde{\mathbf{z}}_t) + \sum_{t} \sum_{s \in \mathcal{N}_S^k(t)} \mathcal{D}(\tilde{\mathbf{z}}_t, \mathbf{x}_s)$$

$\mathbf{x}_s$'s *k* neighbors in the target domain

$\tilde{\mathbf{z}}_t$'s *k* neighbors in the source domain

Formulation and Optimization of ASC

$$\min_{\mathbf{\Theta}} \sum_{s=1}^{S} \sum_{t \in \mathcal{N}_{\mathcal{T}}^{k}(s)} \mathcal{D}(\mathbf{x}_s, [\mathbf{z}_{t,1:K'}; \mathbf{f}_{\mathbf{\Theta}}(\mathbf{z}_t)]) + \sum_{t=1}^{T} \sum_{s \in \mathcal{N}_{\mathcal{S}}^{k}(t)} \mathcal{D}([\mathbf{z}_{t,1:K'}; \mathbf{f}_{\mathbf{\Theta}}(\mathbf{z}_t)], \mathbf{x}_s) + \lambda \|\mathbf{\Theta}\|_2^2$$

regularization term

non-smooth in $\mathbf{\Theta}$, because neighbors are dependent on $\mathbf{\Theta}$

Alternating Optimization (EM-like algorithm):

- Fix $\mathbf{\Theta}$, update neighbors $\mathcal{N}_{\mathcal{T}}^{k}(s)$ and $\mathcal{N}_{\mathcal{S}}^{k}(t)$
- Fix neighbors $\mathcal{N}_{\mathcal{T}}^{k}(s)$ and $\mathcal{N}_{\mathcal{S}}^{k}(t)$, update $\mathbf{\Theta}$

Evaluation in BRASS Project Phase 1

Weather Underground Data
13 geographical clusters, each with 3 stations

Sensor change: an individual sensor is replaced by the same sensor at a nearby station from the same cluster

Ref error: average signal difference of a particular sensor over a cluster

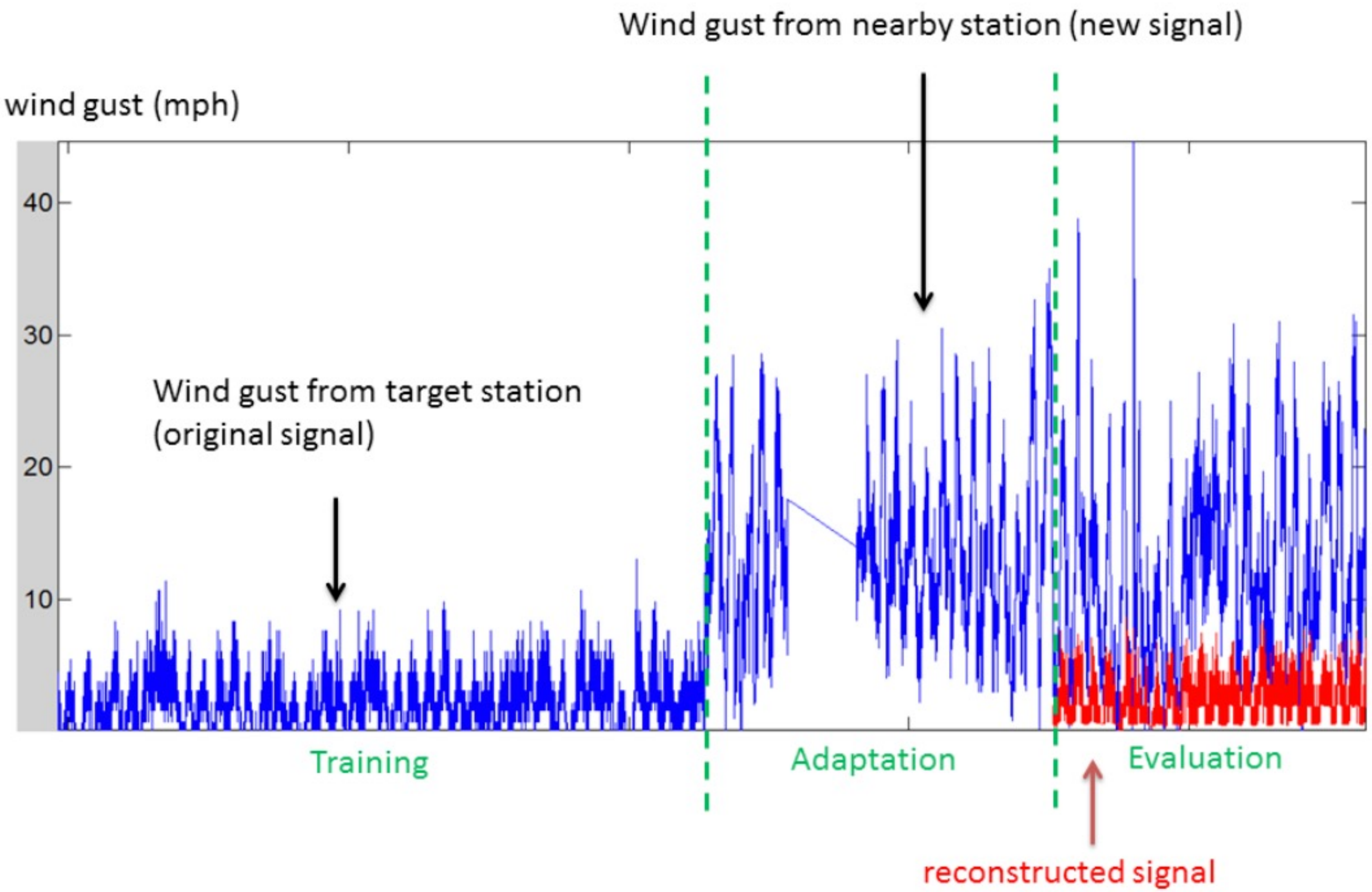
Success: reconstruction error < ref error

5 individual sensors from a station (sample rate: every 5-10 mins)

ID	Type	Unit
1	Temperature	℃
2	Dew point	℃
3	Humidity	%
4	Wind speed	mph
5	Wind gust	mph

Sensor	Success rate (%)	Avg. Imp. over ref error (%)
temperature	95.4	61.6
humidity	96	65.8
dew point	100	71.1
wind speed	84.6	28.7
wind gust	66.7	24.0

Evaluation in BRASS Project Phase 1



Results on UUV Data

A UUV travels from a starting point to an end point in a simulated environment

Sensors: propeller RPM, Waterspeed, DVL (surge, heave, sway, pitch, roll, depth and heading)

Replaced sensor: surge/heave/sway
New sensor: biased version of surge/heave/sway
Reference sensors: remaining sensors

ASC achieves an average improvement of 8.83% over the competing methods

Directly using the new sensor

Adaptation using the reference sensors only

Adaptation using the reference sensors and "guessed" new sensors

Sensor	**Replace**	**Refer**	**ReferZ**	**ASC**	Imp.(%)
surge	2.47	0.66	0.58	0.47	18.9
heave	0.13	0.020	0.020	0.019	6.5
sway	2.31	0.74	0.72	0.71	1.1

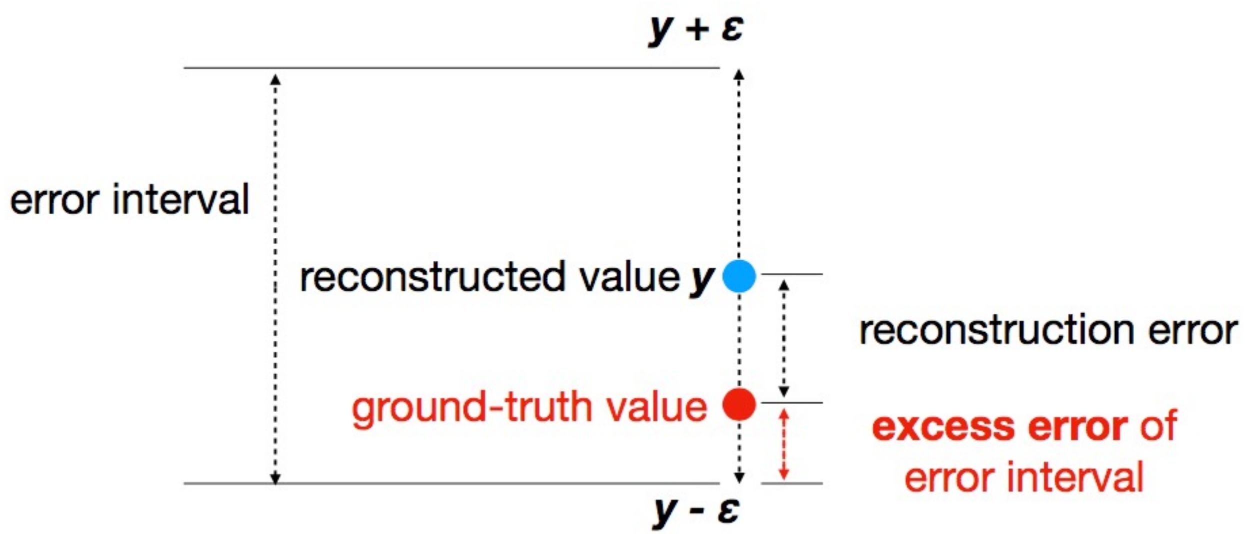
Reconstruction errors (RMSE) averaged over 20 simulated trips

Further Improvement on ASC

- Estimating Adaptation Quality
- Exploiting Many Sensors
- Leveraging Spatial and Temporal Information

Estimating Adaptation Quality

- Useful for Upper-layer Software
- Minimizing the "excess error"
- Estimated from similar data points in the source domain



Exploiting Many Sensors

A large number of sensors is challenging
- More noise
- Larger parameter space

ASC^{SEL}:
- Selecting a subset of reference sensors more correlated with the replaced sensors
- Selecting a subset of new sensors more correlated with the replaced sensors

Results with over 200 sensors
16.5% average improvement

replaced sensor	Reconstruction Error	
	ASC	**ASC^{SEL}**
temperature (°F)	0.47	0.38
humidity (%)	0.53	0.47
dew point (°F)	0.47	0.44
wind speed (mph)	5.04	4.83
wind gust (mph)	6.28	5.61
pressure (Pa)	3.17	1.68

Leveraging Spatial and Temporal Information

Spatial and temporal information of sensors are often available

Station 1:
34.0°N, 118.4°W

Time	Temp
8:50 AM	24.2
8:55 AM	24.3
9:00 AM	--
9:05 AM	--
9:10 AM	--

1

direct replacement
(closest time)

Station 2:
37.7°N, 122.4°W

Time	Temp
9:02 AM	26.6
9:12 AM	27.5

2

calibration function
based on time and location

learned from historical data
of multiple stations

Outline

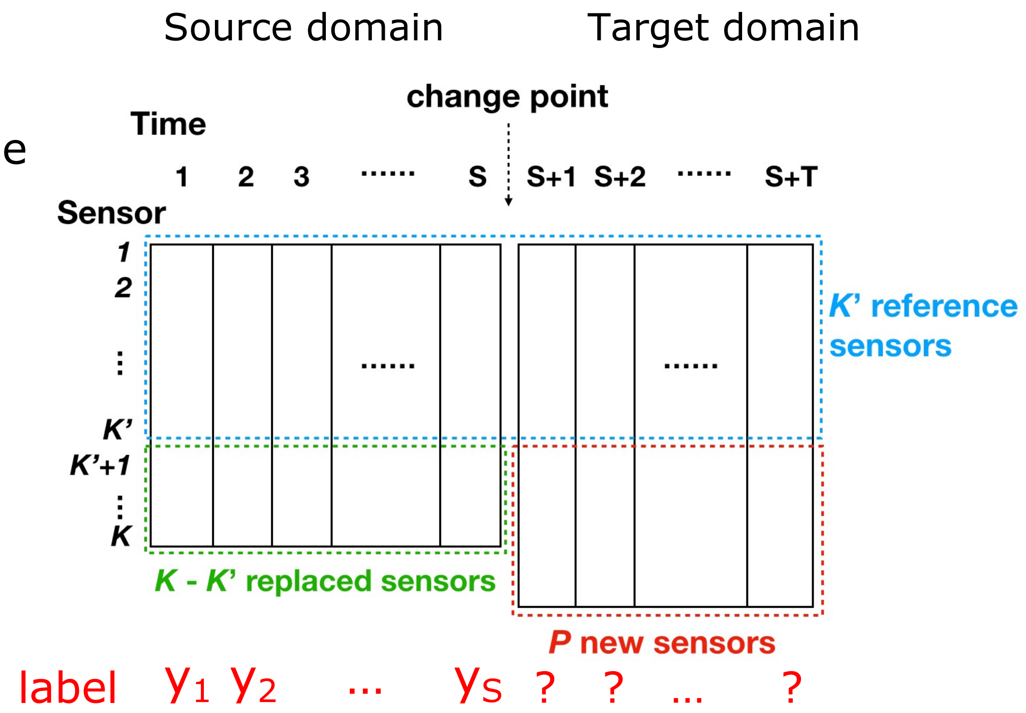
- Sensor-level Adaptation to Sensor Changes
- **Model-level Adaptation to Sensor Changes**
- Joint Detection and Adaptation to Sensor Failures

Model-level Adaptation to Sensor Changes

Domain adaptation (DA)
Goal: optimizing a classifier on the target domain

Unsupervised: no labels on the target domain

Heterogeneous: two domains have different feature representations



Model-level Adaptation to Sensor Changes

Earlier approaches for unsupervised domain adaptation (before [Shi and Sha, '12]): **Two-stage** learning paradigm
1) Identify a **domain-invariant** feature space
2) Build a classifier in that feature space

Issue: the domain-invariant feature space may NOT be **discriminative**: projecting into irrelevant feature dimensions may make two domains look invariant!

Model-level Adaptation to Sensor Changes

Earlier approaches for unsupervised domain adaptation (before [Shi and Sha, '12]): **Two-stage** learning paradigm
1) Identify a **domain-invariant** feature space
2) Build a classifier in that feature space

Issue: the domain-invariant feature space may NOT be **discriminative**: projecting into irrelevant feature dimensions may make two domains look invariant!

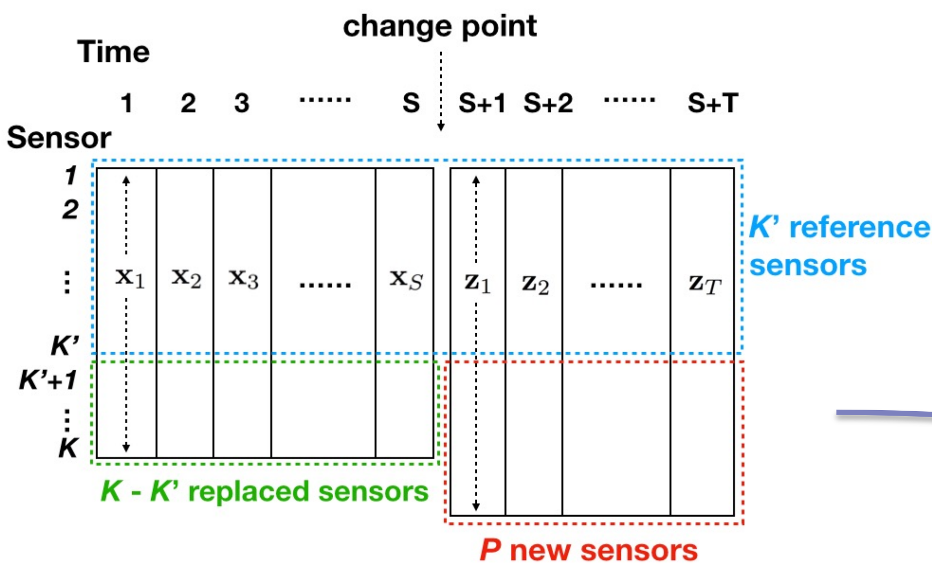
Our approach: One-stage learning
Identify a latent feature space
- **Discriminative** for the target domain
- **Domain-invariant**

Model-level Adaptation to Sensor Changes

Nearest-neighbor-based formulation

Better than competing methods on object recognition, sentiment analysis [Shi and Sha, '12] and weather classification tasks

Source domain Target domain



y_1 y_2 ... y_S

$$\mathbf{v}_t = \mathbf{B}\mathbf{z}_t$$

transformation matrices to learn

$$\mathbf{v}_s = \mathbf{L}\mathbf{x}_s$$

latent feature space

V

Discriminative for the target domain

Domain-invariant

Outline

- Sensor-level Adaptation to Sensor Changes
- Model-level Adaptation to Sensor Changes
- **Joint Detection and Adaptation to Sensor Failures**

Joint Detection and Adaptation to Sensor Failures

- **Constraint-based** framework: determine the **reconstruction relationships** among sensors and express them into **constraints**

$$(\text{temperature} - \underbrace{f}_{\text{reconstruction function}}(\text{dew point}, \text{humidity})) \leq \underbrace{\epsilon^2}_{\text{reconstruction error}}$$

Joint Detection and Adaptation to Sensor Failures

- **Constraint-based** framework: determine the **reconstruction relationships** among sensors and express them into **constraints**

$$\underbrace{(\text{temperature} - f(\text{dew point}, \text{humidity}))}_{} \leq \epsilon^2$$

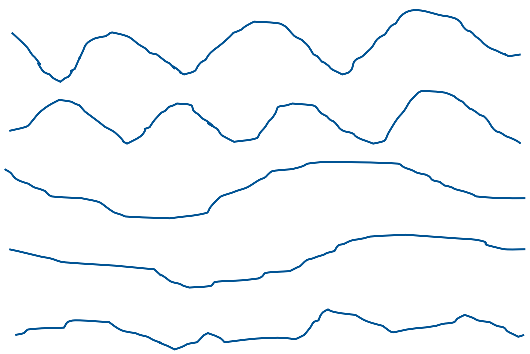
reconstruction function   reconstruction error

- When new sensor values come in:
  - **Detection**: check violated constraints and infer failed sensors
  - **Adaptation**: reconstruct failed sensor values by using relevant constrain

Training Phase: Learning the Constraints Among Sensors

Historical sensor data
(all sensors work properly)

S_1
S_2
S_3
S_4
S_5



Extract sensor relationships into constraints

Constraint Set

$(S_3 + 3S_4^2 - 4.6S_5 - 0.5)^2 < 0.2^2$

$(S_1 - 2\log(S_3) + 2.3)^2 < 0.1^2$

$(S_1 - 3.5)^2 < 0.8^2$

……

Encoding nonlinear reconstruction relationships among sensors

USC University of Southern California

Testing Phase: Using Constraints for Detection and Adaptation

New sensor values at time t

Constraint Set

S_1	5.63
S_2	1.91
S_3	3.42
S_4	70.89
S_5	3.16

Detection

1. Find violated constraints

2. Identify minimum set of failed sensors by solving a special Integer Linear Program (ILP)

Testing Phase: Using Constraints for Detection and Adaptation

New sensor values at time t

S_1	5.63
S_2	1.91
S_3	3.42
S_4	70.89
S_5	3.16

Constraint Set

Detection

1. Find violated constraints

2. Identify minimum set of failed sensors by solving a special Integer Linear Program (ILP)

v_k = 1 if sensor *k* fails, otherwise 0

Testing Phase: Using Constraints for Detection and Adaptation

New sensor values at time t

Constraint Set

S_1	5.63
S_2	1.91
S_3	3.42
S_4	70.89
S_5	3.16

Detection

1. Find violated constraints

2. Identify minimum set of failed sensors by solving a special Integer Linear Program (ILP)

v_k = 1 if sensor *k* fails, otherwise 0

If a constraint is violated, then *at least* one sensor involved in the constraint fails:

$$\sum_{k \text{ in the constraint}} v_k \geq 1$$

Testing Phase: Using Constraints for Detection and Adaptation

New sensor values at time t

Constraint Set

S_1	5.63
S_2	1.91
S_3	3.42
S_4	70.89
S_5	3.16

Detection

1. Find violated constraints
2. Identify minimum set of failed sensors by solving a special Integer Linear Program (ILP)

v_k = 1 if sensor *k* fails, otherwise 0

If a constraint is violated, then *at least* one sensor involved in the constraint fails:

$$\sum_{k \text{ in the constraint}} v_k \geq 1$$

Objective: $$\min \sum_{k\in[1,K]} v_k$$

0-1 Integer Linear Program

Testing Phase: Using Constraints for Detection and Adaptation

Constraint Set

New sensor values at time t

S_1 5.63
S_2 1.91
S_3 3.42
S_4 70.89
S_5 3.16

Detection

1. Find violated constraints
2. Identify minimum set of failed sensors by solving a special Integer Linear Program (ILP)

Testing Phase: Using Constraints for Detection and Adaptation

New sensor values at time t

S_1 5.63
S_2 1.91
S_3 3.42
S_4 70.89
S_5 3.16

Constraint Set

Detection

1. Find violated constraints

2. Identify minimum set of failed sensors by solving a special Integer Linear Program (ILP)

Adaptation

Reconstruct failed sensor values from the working sensors using the most relevant constraint

Testing Phase: Using Constraints for Detection and Adaptation

Constraint Set

New sensor values at time t

S_1 5.63
S_2 1.91
S_3 3.42
S_4 70.89
S_5 3.16

Detection

1. Find violated constraints

2. Identify minimum set of failed sensors by solving a special Integer Linear Program (ILP)

Adaptation

Reconstruct failed sensor values from the working sensors using the most relevant constraint

S_3 fails, S_4 and S_5 work properly

$(S_3 + 3S_4^2 - 4.6S_5 - 0.5)^2 < 0.2^2$

$S_3 \longleftarrow -3S_4^2 + 4.6S_5 + 0.5$

Learning Constraints From Historical Data

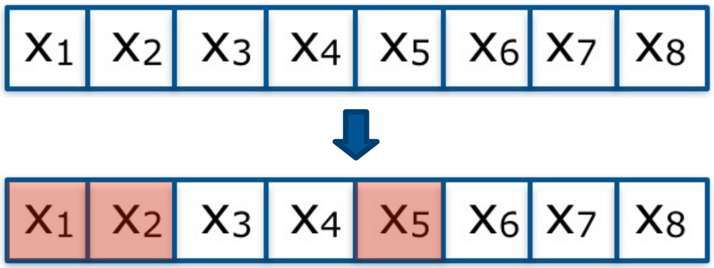
- Properties of desired reconstruction relationships
  - Accuracy: low reconstruction error
  - Comprehensiveness: capturing various types of relationships
  - Compactness: small # of sensors involved; small # of learned constraints

Learning Constraints From Historical Data

- Iteratively grouping sensors into “disjoint” subsets and learn reconstruction functions within subsets

Input sensors:

Target sensors: y



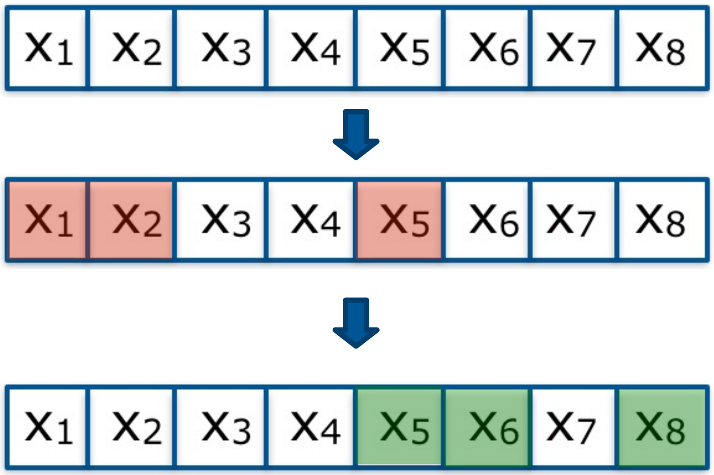
$$y = f(x_1, x_2, x_5)$$

Learning Constraints From Historical Data

- Iteratively grouping sensors into “disjoint” subsets and learn reconstruction functions within subsets

Input sensors:

Target sensors: y



$y = f(x_1, x_2, x_5)$

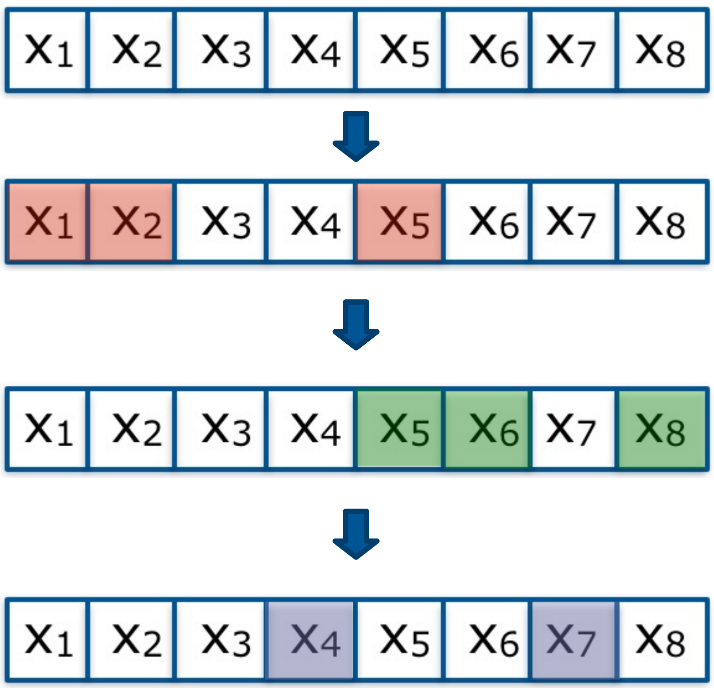
$y = g(x_5, x_6, x_8)$

Learning Constraints From Historical Data

- Iteratively grouping sensors into "disjoint" subsets and learn reconstruction functions within subsets

Input sensors:

Target sensors: y



$y = f(x_1, x_2, x_5)$

$y = g(x_5, x_6, x_8)$

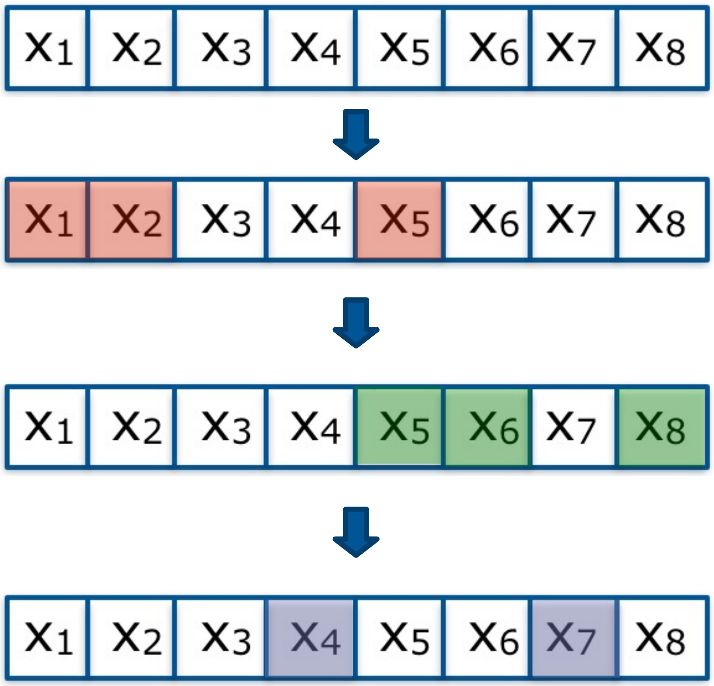
$y = h(x_4, x_7)$

Learning Constraints From Historical Data

- Iteratively grouping sensors into “disjoint” subsets and learn reconstruction functions within subsets

Input sensors:

Target sensors: y



selecting vector $\mathbf{w}_1$

selecting vector $\mathbf{w}_2$

Selecting “disjoint” subsets

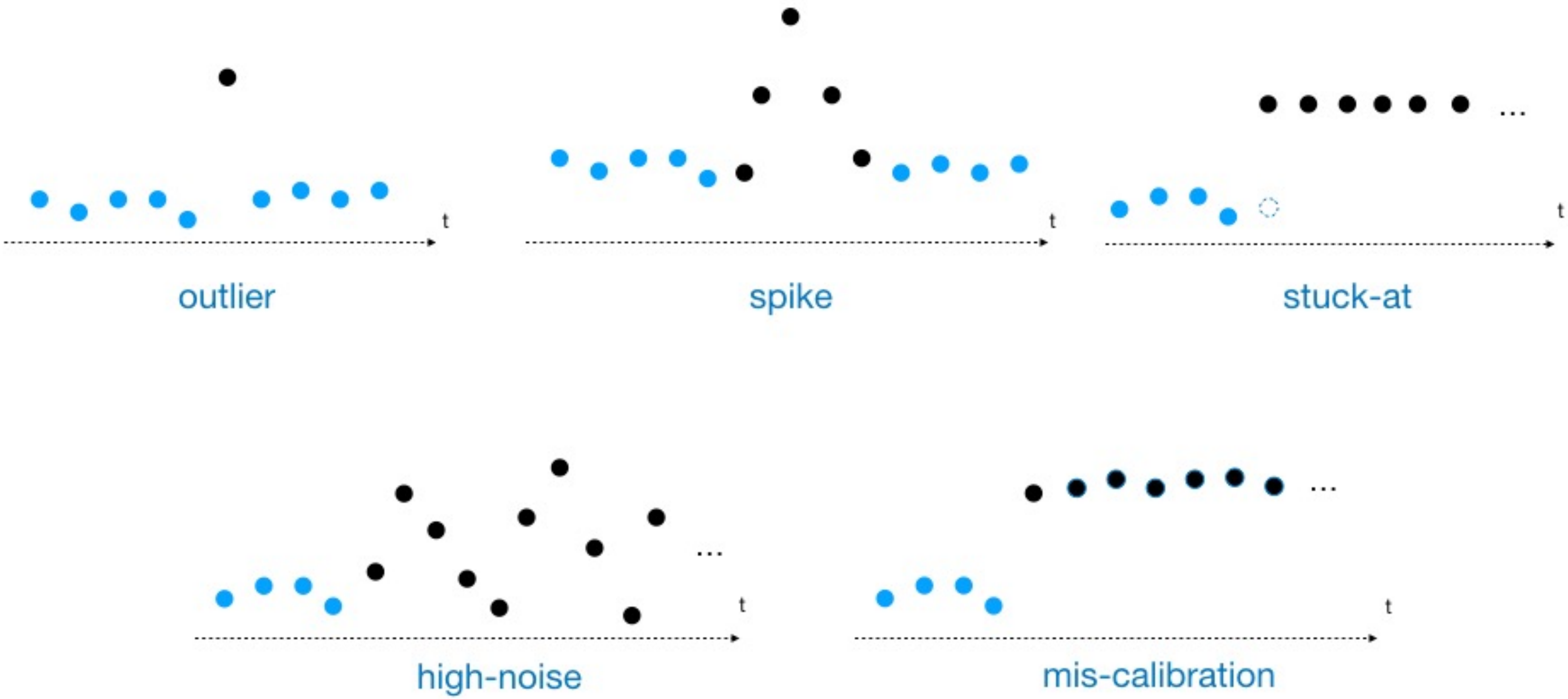
$$\min_{\mathbf{w}_3} \mathbb{E}\left[(y - \mathbf{w}_3{}^T\mathbf{x})^2\right] + \lambda|\mathbf{w}_3{}^T\mathbf{w}_1| + \lambda|\mathbf{w}_3{}^T\mathbf{w}_2|$$

Reducing reconstruction error

LASSO Problem

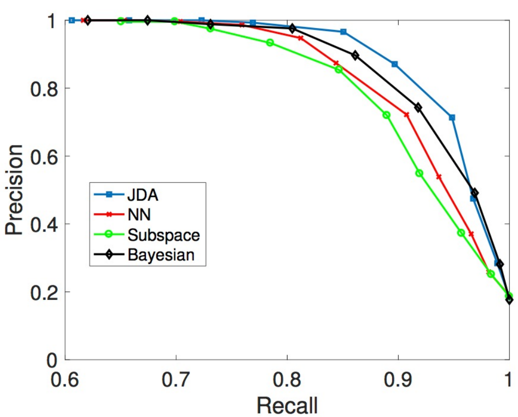
Identifying Failure Modes

- Five common modes: Outlier, Spike, Stuck-at, High-noise, Mis-calibration

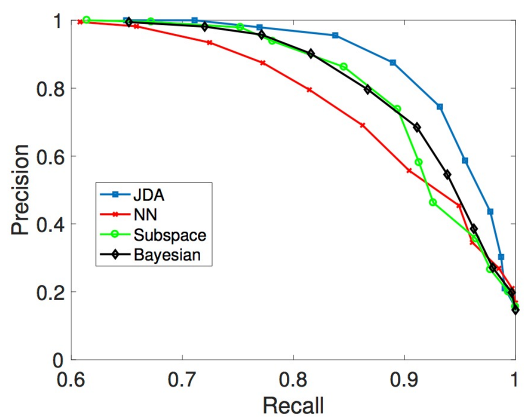


Results on Weather Data

- Five modes of sensor failures are simulated (multi-sensor failures involved)
- Our approach (JDA) achieves higher detection rates and lower reconstruction errors
  - more significant on sensor values with smaller variances



(a) Temperature



(b) Humidity

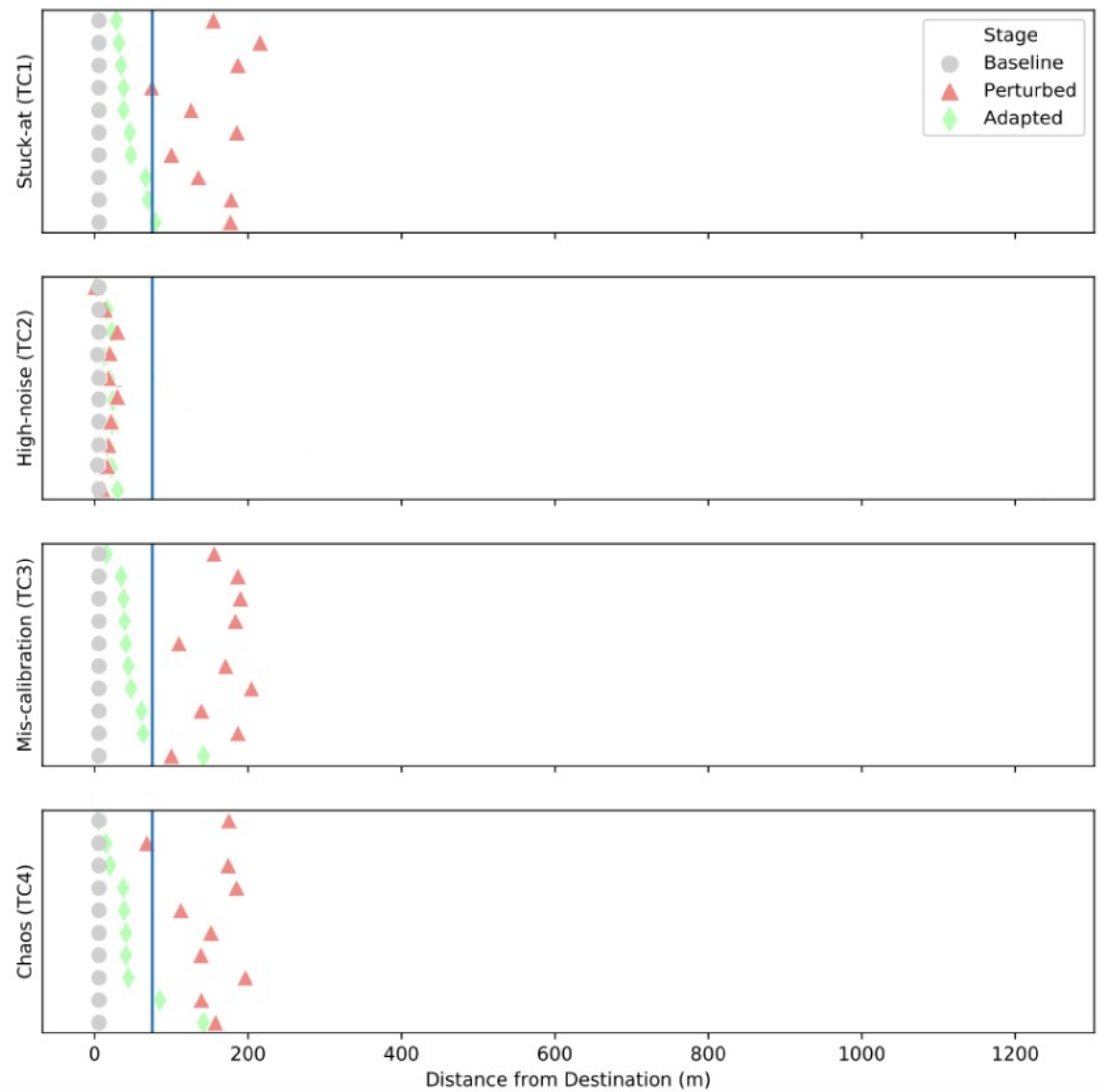
Sensor	Reconstruction Error	
	Reference	**JDA**
Temperature	1.32	0.23
Humidity	5.28	0.41
Dew Point	1.15	0.33
Wind Speed	5.36	3.81
Wind Gust	5.12	3.68
Pressure	3.71	2.25

Results on Austin weather stations

Evaluation in BRASS Project Phase 2: UUV Results

USC University of Southern California

- A UUV travels to a destination
- Perturbations are simulated to affect the UUV's ability to localize its position
- PASS: if the UUV is less than 75 meters from the destination
- Our system achieves PASS in 90% of the cases



Related Work

- **Detecting Sensor Failures and Changes**
  - Change point detection [Aminikhanghahi and Cook '16] [Pimentel et al., '14]
    - Distribution-based [Kawahara and Sugiyama, '12] [Harchaoui et al., '09] [Yamanishi and Takeuchi, '02]
    - Reconstruction-based [Crook et al., '02] [Singh and Markou, '04] [Ide and Tsuda, '07] [Chatzigiannakis et al., '06]
    - Probabilistic [Adams and MacKay, '07] [Saatci et al., '10] [Dereszynski and Dietterich, '12] [Dietterich et al. '12]
    - Distance-based [Angiulli and Pizzuti, '02] [Bay and Schwabacher, '03] [Chawla and Sun, '06] [Keogh et al., '01] [Ide et al., '13] [Budalakoti et al., '06] [Chen et al., '15]

Our detection approach explores multiple nonlinear relationships among sensors, and can potentially detect sensor changes with significantly higher accuracy

- **Reconstruction of Sensor Readings**
  - Most detection methods do not address how to automatically recover
  - Some probabilistic methods [Dereszynski and Dietterich, '12] [Dietterich et al. '12] can be used to reconstruct changed sensor, but cannot leverage new sensors
  - FFX [McConaghy '11] is applied to extract sensor-specific transformations

Our sensor-level adaptation approach can adapt to new sensors, which are not possible by existing approaches

Related Work

- **Domain Adaptation** [Pan and Yang, ‘10]
  - Unsupervised domain adaptation
    - Two-stage learning paradigm: domain invariant, then discriminative [Pan et al., ‘11] [Gopalan et al., ‘11] [Gong et al., ‘12] [Chen et al., ‘12] [Shimodaira, ‘00] [Bickel et al., ‘07] [Huang et al., ‘07] [Blitzer et al., ‘06] [Glorot et al., ‘11]
    - One-stage learning paradigm: discriminative + domain invariant [Csurka et al., ‘16] [Baktashmotlagh et al., ‘13] [Baktashmotlagh et al., ‘14] [Tzeng et al., ‘15] [Ganin et al., ‘16]
  - Heterogeneous domain adaptation
    - Domain invariant feature space [Kulis et al., ‘11] [Wang and Mahadevan, ‘11] [Argyriou et al., ‘08] [Duan et al., ‘12] [Shi et al., ‘10] [Harel and Mannor, ‘10] [Wei and Pal, ‘11] [Yeh et al., ‘14] [Chen et al., ‘16]
    - Sample-correspondence between domains [Dai et al., ‘08] [Socher et al., ‘13] [Zhou et al., ‘14]
    - Feature correlations [Zhao and Hoi, ‘10] [Hou and Zhou, ‘16]
  - Domain adaptation on time-series data [Purushotham et al., ‘17]

Our sensor-level adaptation approach does not require labels in the target domain. Our model-level adaptation approach is based on our publication [Shi and Sha, ‘12], which proposed the one-stage learning paradigm and enabled direct optimization of classifiers on the target domain.

Conclusions

- **Sensor-level adaptation approaches for sensor failures and changes**
  - Adapting to new sensors
  - Estimating the quality of adaptation
  - Leveraging sensor-specific transformations as well as spatial and temporal information
- **Model-level adaptation approach for sensor failures and changes**
  - One-stage domain adaptation that is unsupervised and heterogeneous
- **Constraint-based framework for joint detection and adaptation to sensor failures**
  - Detecting and adapting to multi-sensor failures
- **Validated on sensor data from the weather and UUV domains (BRASS Evaluation)**
- **Future work:** applying to large-scale sensor data; integration into survivable software systems

Thank You!

Question?

- Why reconstructing replaced sensors than using new sensors directly?

- Show General applicability
- Domains: Image recognition and sentiment analysis

- More diagrams. Significant impact

Results on Weather Data

Weather Underground Data

30 weather stations from 10 geographical clusters

Random triplet: station A1, A2 from one cluster, B from another

Replaced sensor: a sensor from A1
New sensor: the same sensor from A2
Reference sensors: remaining sensors from A1 and all sensors from B

2016 data for training, 2017 data for testing

6 individual sensors from a station

ID	Type	Unit
1	Temperature	℃
2	Dew point	℃
3	Humidity	%
4	Wind speed	mph
5	Wind gust	mph
6	Pressure	Pa

Results on Weather Data

Weather Underground Data

Reconstruction errors (RMSE) averaged over random triplets

ASC achieves an average improvement of 6.35% over the competing methods

Directly using the new sensor

Reconstruction function learned on the reference sensors

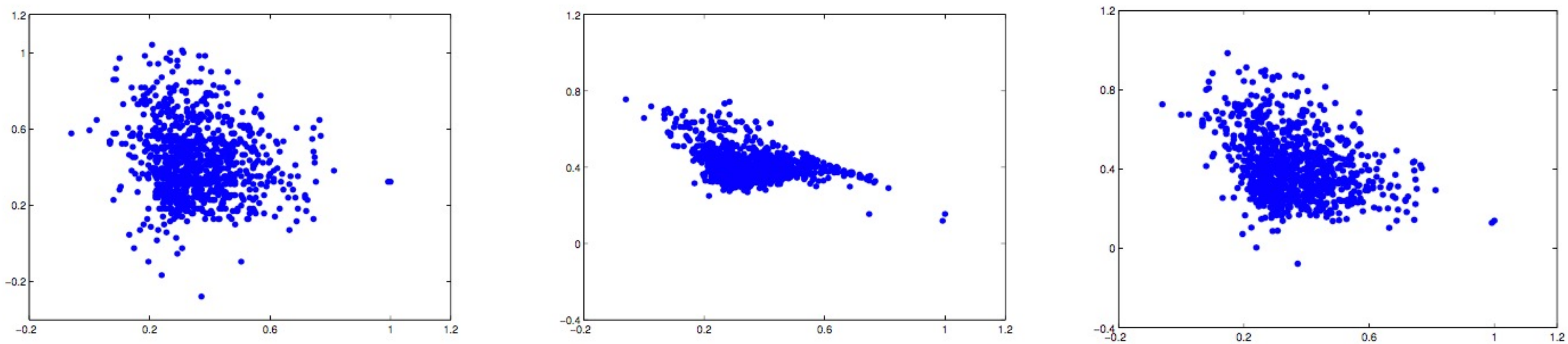
Reconstruction function learned on the reference sensors and estimated new sensors (on the source domain)

Sensor	**Replace**	**Refer**	**ReferZ**	**ASC**	Imp.(%)
temperature	3.94	0.61	0.59	0.57	4.1
humidity	5.73	0.72	0.71	0.72	-1.7
dew point	3.89	0.70	0.68	0.67	2.8
wind speed	8.24	5.20	5.21	5.11	1.7
wind gust	10.81	6.65	6.65	6.31	5.0
pressure	7.82	3.39	2.48	1.83	26.2

Results on Weather Data

Weather Underground Data

Reconstructed pressure



(a) ground truth (b) **Refer** (c) **ASC**

Wind speed