The Thirty-Second International Florida
Artificial Intelligence Research Society Conference (FLAIRS-32)

# Automatic Adaptation to Sensor Replacements

**Yuan Shi**
Information Sciences Institute
University of Southern California
yuanshi@usc.edu

**T. K. Satish Kumar**
Information Sciences Institute
University of Southern California
tkskwork@gmail.com

**Craig A. Knoblock**
Information Sciences Institute
University of Southern California
knoblock@isi.edu

## Abstract

Many software systems run on long-lifespan platforms that operate in diverse and dynamic environments. If these software systems could automatically adapt to hardware changes, it would significantly reduce the maintenance cost and enable rapid upgrade. In this paper, we study the problem of how to automatically adapt to sensor changes, as an important step towards building such long-lived, survivable software systems. We address the adaptation scenarios where a set of sensors are replaced by new sensors. Our approach reconstructs sensor values of replaced sensors by preserving distributions of sensor values before and after the sensor change, thereby not warranting a change in higher-layer software. Compared to existing work, our approach has the following advantages: a) exploiting new sensors without requiring an overlapping period of time between new sensors and old ones; b) providing an estimation of adaptation quality; and c) scaling to a large number of sensors. Experiments on weather data and Unmanned Undersea Vehicle (UUV) data demonstrate that our approach can automatically adapt to sensor changes with higher accuracy compared to baseline methods.

## Introduction

An increasing number of applications require long-term autonomy of software systems and their capability to operate in dynamic environments. Maintaining the quality, durability and performance of these software systems is very challenging and labor-intensive. Failure to effectively or promptly adapt to hardware and resource changes can result in technically inferior and potentially vulnerable systems (Hughes et al. 2016). If software systems could automatically adapt to these changes, the time and effort required for maintenance would be significantly reduced. As an important step towards building long-lasting survivable software systems, we study the problem of how to automatically adapt to sensor changes in which a set of sensors are replaced by new sensors. Solutions to this problem can have a broader impact given that an increasing number of sensors are deployed in real-world applications (Dereszynski and Dietterich 2011; Gubbi et al. 2013).

Sensor changes often occur in real-world systems due to replacement of failed sensors, sensor upgrade, energy optimization, etc. (Tong et al. 2011; Lai et al. 2012; Hughes et al. 2016). Throughout this paper, we refer to sensors that are replaced by new sensors as *replaced sensors*. When sensor change occurs, sensor values from new sensors may not match those from replaced sensors. For example, miscalibration could exist between replaced sensors and new sensors even when they measure the same types of signals. Furthermore, new sensors may measure additional types of signals that do not exist before the sensor change. To automatically adapt to sensor changes, we would like to reconstruct sensor values of replaced sensors using the remaining sensors and new sensors. Such reconstruction is possible based on the observation that sensor values are often correlated in real-world systems (Elnahrawy and Nath 2004). Taking weather sensors as an example, temperature, dew point and humidity are highly correlated and each sensor value can be efficiently reconstructed from the other two (Alduchov and Eskridge 1996). Existing literature on sensor changes mainly focuses on change detection, but rarely addresses how to adapt to these changes (Basseville, Nikiforov, and others 1993; Gustafsson and Gustafsson 2000; Brodsky and Darkhovsky 2013; Aminikhanghahi and Cook 2016). Typically, human experts are required to examine and respond to detected changes.

One approach to learning such a reconstruction function is to simply ignore new sensors, and reconstruct replaced sensors using the remaining ones. Assuming that we have access to sufficient historical sensor values of these sensors, such a reconstruction function can be learned straightforwardly via classical regression methods (Friedman, Hastie, and Tibshirani 2001). However, new sensors may contain complementary information over the remaining sensors, which may help us better reconstruct replaced sensors. As an extreme example, if new sensors are exactly the same as replaced sensors, using their sensor values definitely aids reconstruction.

Learning a reconstruction function that exploits new sensors poses unique challenges, since there is no overlapping period of time between the replaced and new sensors. Classical regression methods cannot be directly applied. To address this challenge, we propose an approach called **ASC** (Adaptation to Sensor Changes) that learns a reconstruction function to preserve sensor value distributions before and after the sensor change. We further improve **ASC** in two aspects motivated by real-world applications. First, we develop a procedure to deal with a large number of sensors by selecting a subset of important sensors. This procedure can significantly reduce the overfitting to noisy values as well as the overall computational cost. It enables our approach to continuously exploit new sensors in an open environment. Second, we propose a method to dynamically estimate the adaptation quality, which enables higher-layer software components to determine whether or not to accept an adaptation. For empirical study, we evaluate **ASC** on sensor data from weather and UUV domains. In most of the evaluation cases, **ASC** outperforms other baseline approaches, achieving an average improvement of 5.7%.

## Approach

We study the general setting of sensor change in the context of a compound sensor. Imagine we have a compound sensor consisting of multiple individual sensors. For example, a compound sensor can be a weather station containing several weather sensors measuring temperature, dew point, wind speed, etc. An instant of time at which some sensors are replaced by new sensors is called a sensor change point.[1] We consider two general scenarios:

- *Individual sensor change*, where only a subset of sensors in the compound sensor is replaced;
- *Compound sensor change*, where the entire compound sensor is replaced. This happens in practice when sensor replacement is conducted in the compound level.

For individual sensor change, since there is no overlapping period of time between replaced sensors and new sensors, the remaining sensors are the key to link the two. In the following, we call them as *reference sensors*. Intuitively, if reference sensors are correlated with both replaced sensors and new sensors, they can be helpful for reconstructing replaced sensors from new sensors. For compound sensor change, however, there are no reference sensors from the compound sensor because all sensors are replaced. In this scenario, adaptation to new sensors is very challenging or even impossible. To enable reasonable adaptation, we assume that we have access to some reference sensors outside the compound sensor. For example, in the context of weather stations, we can use sensors in other stations as reference sensors. With the notion of reference sensors, the two scenarios can be viewed in a unified way: reference sensors always work properly, and replaced sensors are replaced by new sensors at the change point. Fig. 1 visualizes this unified view and the corresponding notations (explained below).

**Notations.** Suppose we are given $K$ individual sensors, among which $K'$ sensors are reference sensors. We assume that

- All sensors generate sensor values at fixed time intervals, and sensor values are temporally aligned;
- Sensor values start at time 1. At time $S+1$, $K-K'$ sensors are replaced by $P$ new sensors. We have sensor values until time $S+T$;

[1]This paper only addresses a single change point. Extension to multiple change points is straightforward.

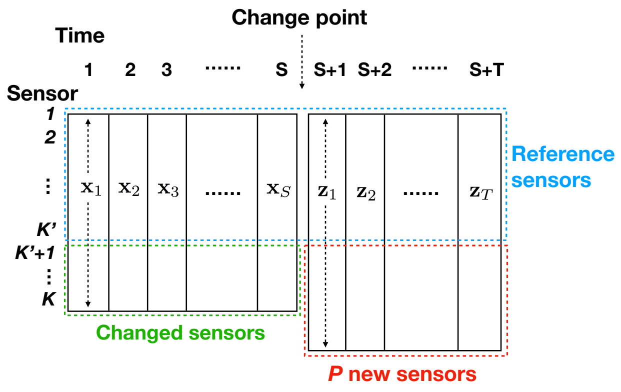

Figure 1: Sensor change setting and notations.

- There is only a single change point, and it is already given (detecting the change point is not the focus of this paper).

Let $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_S$ be sensor values before the sensor change, where $\mathbf{x}_s \in \mathcal{R}^K$ represents sensor values at time $s \in \{1, 2, \cdots, S\}$, and $\mathbf{x}_{s,k}$ represents the corresponding sensor value from sensor $k \in \{1, 2, \cdots, K\}$. Additionally, let us assume that at time $S+1$, sensors $K'+1, \cdots, K$ are replaced by $P$ new sensors. Let $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_T$ denote sensor values after the sensor change, where $\mathbf{z}_t \in \mathcal{R}^{K'+P}$ represents sensor values at time $S+t\,(t \in \{1, 2, \cdots, T\})$. Note that we use $s$ to index $\mathbf{x}$ and $t$ to index $\mathbf{z}$. Based on this setting, $\{\mathbf{x}_{s,k}\}$ and $\{\mathbf{z}_{t,k}\}$ $(k \in \{1, 2, \cdots, K'\})$ represent sensor values of reference sensors, and $\{\mathbf{z}_{t,k}\}$ $(k \in \{K'+1, K'+2, \cdots, K'+P\})$ represent sensor values of the $P$ new sensors. Fig. 1 illustrates the above notations.

**Assumptions and Intuition.** Our approach reconstructs sensor values of replaced sensors from time $S+1$ to $S+T$ based on reference sensors and new sensors. The underlying assumptions are:

- Sensor values from reference sensors are correlated with those from replaced sensors;
- Sensor values from reference sensors are correlated with those from new sensors.

Such assumptions typically hold in real-world systems because sensor values of different sensors are often correlated (Elnahrawy and Nath 2004).

Our approach is based on the following intuition. New sensors may contain complementary information over reference sensors, useful for reconstructing replaced sensors. Fig. 2 illustrates this intuition, where the reference sensor, replaced sensor, and the new sensor are temperature, humidity and dew point, respectively. The left plot shows two selected samples from historical data. We can see that for the same temperature value, humidity can take different values. The middle plot shows that if we attempt to reconstruct humidity from temperature alone, via the $g$ function, then the reconstructed humidity values become exactly the same, since the temperature information alone is insufficient for

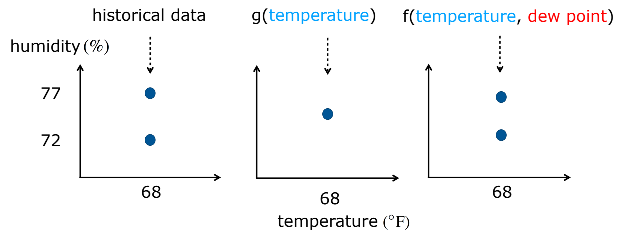

Figure 2: Illustration of the intuition behind our approach.

the reconstruction. The right plot shows that by incorporating dew point as a new signal, the reconstructed humidity values are distributed similar to those in the left plot. This is expected because dew point contains complementary information over temperature for reconstructing humidity. The above intuition leads to the key idea of our approach: to learn a reconstruction function that preserves the sensor value distributions before and after the sensor change.

**Formulation.** In the following, we refer to sensor values before the sensor change as the *source domain*, and sensor values after the sensor change as the *target domain*. Similar notions are used in domain adaptation and transfer learning communities (Pan and Yang 2010). Specifically, we aim to learn a reconstruction function $\mathbf{f}_{\mathbf{\Theta}}(\mathbf{z})$ that maps sensor values after the sensor change to values before the sensor change, where $\mathbf{\Theta}$ denotes the parameters of the function. Note that the output of $\mathbf{f}_{\mathbf{\Theta}}(\mathbf{z})$ is a matrix when there are more than one replaced sensors. In our implementation, we use the form

$$\mathbf{f}_{\mathbf{\Theta}}(\mathbf{z}) = \mathbf{\Theta}^{\top}\mathbf{h}(\mathbf{z}) \quad (1)$$

where $\mathbf{h}()$ is a nonlinear feature mapping, e.g., quadratic features or nonlinear features derived from neural networks.

We are interested in $\mathbf{f}_{\mathbf{\Theta}}(\mathbf{z})$ such that distributions are similar across domains after the reconstruction. This motivates us to seek $\mathbf{f}_{\mathbf{\Theta}}(\mathbf{z})$ such that the two sets of samples $\{\mathbf{x}_s\}$ and $\{[\mathbf{z}_{t,1:K'}; \mathbf{f}_{\mathbf{\Theta}}(\mathbf{z}_t)]\}$ (i.e., reconstructed samples in the target domain)[2] are "mixed" as much as possible. When this happens, each source-domain sample $\mathbf{x}_s$ becomes close to its $k$-nearest neighbors in the target domain, and vice versa. Therefore we propose the following objective function to minimize the cross-domain $k$-nearest neighbor distances

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

where $\mathcal{D}(\cdot,\cdot)$ is the distance function defined in the space $\mathbf{x} \in \mathcal{R}^K$. $\mathcal{N}_{\mathcal{T}}^{k}(s)$ denotes the set of indices corresponding to $\mathbf{x}_s$'s $k$-nearest neighbors in the target domain, and $\mathcal{N}_{\mathcal{S}}^{k}(t)$ denotes the set of indices corresponding to $[\mathbf{z}_{t,1:K'}; \mathbf{f}_{\mathbf{\Theta}}(\mathbf{z}_t)]$'s $k$-nearest neighbors in the source domain. Here, nearest neighbors are determined based on the distance function $\mathcal{D}$. $\|\mathbf{\Theta}\|_2^2$ is the regularization term on $\mathbf{\Theta}$ with $\lambda \geq 0$ as the regularization parameter.

For simplicity, we set $\mathcal{D}$ to be the squared Euclidean distance[3]

$$\mathcal{D}(\mathbf{x}_s, [\mathbf{z}_{t,1:K'}; \mathbf{f}_{\mathbf{\Theta}}(\mathbf{z}_t)]) = \|\mathbf{x}_{s,1:K'} - \mathbf{z}_{t,1:K'}\|_2^2 + \|\mathbf{x}_{s,K'+1:K} - \mathbf{f}_{\mathbf{\Theta}}(\mathbf{z}_t)\|_2^2 \quad (3)$$

Letting $v_{st}^2 = \|\mathbf{x}_{s,1:K'} - \mathbf{z}_{t,1:K'}\|_2^2$, we can write (2) as

$$\min_{\mathbf{\Theta}} \sum_{s=1}^{S} \sum_{t \in \mathcal{N}_{\mathcal{T}}^{k}(s)} \left(v_{st}^2 + \|\mathbf{x}_{s,K'+1:K} - \mathbf{f}_{\mathbf{\Theta}}(\mathbf{z}_t)\|_2^2\right) + \sum_{t=1}^{T} \sum_{s \in \mathcal{N}_{\mathcal{S}}^{k}(t)} \left(v_{st}^2 + \|\mathbf{x}_{s,K'+1:K} - \mathbf{f}_{\mathbf{\Theta}}(\mathbf{z}_t)\|_2^2\right) + \lambda\|\mathbf{\Theta}\|_2^2 \quad (4)$$

In Eq. (4), $\mathcal{N}_{\mathcal{T}}^{k}(s)$ and $\mathcal{N}_{\mathcal{S}}^{k}(t)$ are dependent on $\mathbf{\Theta}$, making Eq. (4) non-smooth and non-convex in $\mathbf{\Theta}$. Hence we develop an optimization algorithm that alternates between the following two steps until convergence to a local minimum of Eq. (4):

1. Fix $\mathbf{\Theta}$, update $\mathcal{N}_{\mathcal{T}}^{k}(s)$ and $\mathcal{N}_{\mathcal{S}}^{k}(t)$ based on nearest neighbor search across domains;
2. Fix $\mathcal{N}_{\mathcal{T}}^{k}(s)$ and $\mathcal{N}_{\mathcal{S}}^{k}(t)$, optimize $\mathbf{\Theta}$. When $\mathbf{f}_{\mathbf{\Theta}}(\mathbf{z}_t)$ is linear in $\mathbf{\Theta}$, $\mathbf{\Theta}$ can be solved analytically.

For tuning the regularization parameter $\lambda$, we use a special *leave-one-out* cross-validation strategy. We synthesize a set of sensor change scenarios by treating each sensor in the source domain as the replaced sensor, and using a biased version[4] of that sensor as the new sensor. We then select the optimal $\lambda$ such that the average reconstruction error on these synthesized scenarios is minimized.

## Ability to Exploit Many Sensors

As an increasing number of sensors are deployed in real-world systems, it is crucial for **ASC** to be able to exploit many sensors. This also enables our approach to be deployed in an open environment where new sensors continuously emerge. Dealing with a large number of sensors is challenging in two aspects:

- Noisy sensors are likely to be involved and can degrade adaptation performance. For example, if some reference sensors produce highly noisy values, the nearest neighbor distances can suffer from the noise. Also, noisy values in reference or new sensors can cause the optimization algorithm to get stuck in poor local minima;
- Large number of sensors leads to a large parameter space of $\mathbf{\Theta}$, which significantly increases the computational cost.

[2]We use the notation $1:K'$ to denote a set of indices from 1 to $K'$.

[3]In our implementation, each dimension is normalized into the same scale.

[4]The biased version is created by offsetting each sensor value by the same bias.

In addressing these issues, we develop a two-step procedure to select a subset of useful sensors:

1. Selecting a subset of reference sensors: for each reference sensor, compute the average correlation between its sensor values and those from each replaced sensor, then select $N_{\text{ref}}$ reference sensors with largest average correlation scores;
2. Selecting a subset of new sensors: for each new sensor, compute the average correlation between its sensor values and those from each replaced sensor as well as each selected reference sensor in Step 1, then select $N_{\text{new}}$ new sensors with largest average correlation scores.

Here, $N_{\text{ref}}$ and $N_{\text{new}}$ are set by the user in specific applications. We denote this improved approach as **ASC**$^{\text{SEL}}$.

## Estimating Adaptation Quality

To build survivable software, estimating the quality of adaptation is also important since it enables higher-layer software components to determine whether or not to accept a proposed adaptation. Towards this end, we develop a method to estimate an error interval for the gap between the reconstructed sensor value and the ground truth.

We would like to obtain such an error interval for each reconstructed sensor value and for each sample in the target domain. Given a reconstructed sample in the target domain $[\mathbf{z}_{t,1:K'}; \mathbf{f}_{\Theta}(\mathbf{z}_t)]$, we estimate its error interval for a given reconstructed sensor value from similar samples in the source domain:

1. Find its $\kappa$ nearest neighbors in the source domain according to distances defined in Eq. (3);
2. Compute the standard deviation $\sigma$ on the given reconstructed sensor value among the $\kappa$ neighbors found in Step 1;
3. Set the estimated error interval to be $[-\alpha\sigma, \alpha\sigma]$, where $\alpha > 0$ is a scaling factor. An ideal $\alpha$ makes the error interval as *tight* as possible. $\alpha$ can be tuned on source-domain samples by optimizing the "excess error" notion defined below.

**Excess Error of the Error Interval**. To quantify the tightness of the estimated error interval, we use the notion of *excess error*. It is defined as the gap between the ground-truth value and the closest endpoint of the error interval, when the interval contains the ground-truth value. If the interval does not contain the ground-truth value, we consider the interval invalid. In practice, we can tolerate a small failure rate of the estimated error interval by setting a recall parameter (e.g., 90%). We can then find the smallest $\alpha$ to achieve the given recall and compute the corresponding excess error. Clearly, we favor a smaller excess error as it results in a tighter error interval.

## Related Work

Most existing work on sensor changes focuses on detecting changes (Basseville, Nikiforov, and others 1993; Gustafsson and Gustafsson 2000; Brodsky and Darkhovsky 2013; Aminikhanghahi and Cook 2016; Pimentel et al. 2014; Zhang, Meratnia, and Havinga 2010). These methods rarely address the issue of adaptation, and often rely on human experts to examine and respond to detected changes. Our work, on the other hand, is motivated by the notion of survivable software and aims at automatic adaptation to sensor changes. Although some of the existing detection methods (Dereszynski and Dietterich 2012; Dietterich et al. 2012; Dereszynski and Dietterich 2011) can be used to infer reconstructed sensor values, they are not capable of exploiting new sensors. (Shi and Knoblock 2017) also addresses sensor adaptation but focuses on exploiting previously unseen sensors to improve the underlying learning task. In contrast, our work focuses on adaptation to sensor changes, and it is capable of exploiting a large number of sensors and estimating the quality of adaptation.

Our approach can be viewed as a special case of heterogeneous domain adaptation (Pan and Yang 2010) if we treat sensor values of replaced sensors as labels and sensor values of reference/new sensors as features. However, existing adaptation approaches are not directly applicable to our problem setting.

## Empirical Study

We evaluate **ASC** on sensor data from the weather and UUV domains. We compare **ASC** to three baseline methods:

- **Replace**: non-adaptation method that substitutes each replaced sensor with a new sensor that has closest mean and variance in sensor values.
- **Refer**: adaptation method that reconstructs sensor values of replaced sensors using reference sensors, without exploiting any new sensor.
- **Refer-Z**: adaptation method that works in three steps:
  1. Learn a regression model on the target domain to reconstruct new sensors from reference sensors;
  2. Use the learned regression model to reconstruct new sensors on the source domain;
  3. Learn a reconstruction function on the source domain to reconstruct replaced sensors from reference sensors and reconstructed new sensors.

  This method can work well if new sensors and reference sensors are strongly correlated, which may not hold in real-world applications.

Reconstruction error of each method is measured by RMSE (Root Mean Square Error) between reconstructed sensor values and the ground truth.

### Results on Weather Data

The weather data are collected from Weather Underground[5] which contains a large number of personal weather stations. In our experiments, we use 30 weather stations from 10 geographical clusters. We then generate random triplets across clusters. We generate each triplet in the following way: 1) randomly select two clusters; 2) randomly select two stations from the first cluster (denoted as stations A1 and A2),

[5]https://www.wunderground.com/

and one station from the second cluster (denoted as station B). We use A1 as the compound sensor, A2 as the new sensors, and B as the reference sensors. We generate 100 random triplets, and report averaged results.

Each station consists of six sensors including temperature (°F), humidity (%), dew point (°F), wind speed (mph), wind gust (mph), and pressure (Pa). Sensor values are collected every 5 minutes and are temporally aligned. Sensor values from A1 and A2 are more correlated than those between A1(A2) and B. We use two years of data, with data in 2016 as the source domain and data in 2017 as the target domain.

**Individual sensor changes**. We treat each sensor in A1 as the replaced sensor, the remaining sensors in A1 plus all sensors in B as reference sensors, and all sensors in A2 as new sensors. Table 1 reports reconstruction errors on each replaced sensor, with Imp. (%) showing how much **ASC** improves over the best baseline method. We can see that **ASC** achieves an average improvement of 6.4% over baselines. Only on humidity, **ASC** performs worse than **Refer-Z** but the performance degrade is negligible. This shows the high robustness of **ASC**. In general, **ASC** shows more significant improvement on sensors whose sensor values exhibit large variances (e.g., wind gust and pressure). **Replace** always performs worse than **Refer**, revealing that directly using new sensors can cause significant differences in sensor values. **Refer-Z** improves over **Refer** on 4 cases by leveraging new sensors. **ASC** further improves over **Refer-Z** because it better exploits information from new sensors.

Table 1: Reconstruction errors (RMSE) on weather data for individual sensor changes.

| Sensor | **Replace** | **Refer** | **Refer-Z** | **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 |

**Compound sensor changes**. We treat all sensors in A1 as the replaced sensors, all sensors in B as reference sensors, and all sensors in A2 as new sensors. Table 2 reports reconstruction errors on each replaced sensor separately. **ASC** outperforms baselines in most cases, achieving an average improvement of 5.7%. On wind speed, **ASC** performs slightly worse than **Refer**. Compared to Table 1, **ASC** produces larger reconstruction errors mainly because reference sensors have lower correlations with replaced sensors.

**Dealing with many sensors**. For each triplet, we use A1 as the compound sensor, and simulate reference sensors and new sensors from the remaining 29 stations. Specifically, sensors from 15 randomly selected stations are used as reference sensors, and sensors from the other 14 stations are used as new sensors. This makes the total number of sensors exceed 200.[6] Table 3 reports the results for individual sensor

[6]Some stations have additional types of sensors, e.g., precipitation. Experiments on significantly larger number of sensors remain as future work.

Table 2: Reconstruction errors (RMSE) on weather data for compound sensor changes.

| Sensor | **Replace** | **Refer** | **Refer-Z** | **ASC** | Imp.(%) |
|---|---|---|---|---|---|
| temperature | 3.94 | 0.73 | 0.71 | 0.68 | 4.2 |
| humidity | 5.73 | 0.87 | 0.88 | 0.87 | 0 |
| dew point | 3.89 | 0.75 | 0.74 | 0.72 | 2.6 |
| wind speed | 8.24 | 6.07 | 6.11 | 6.13 | -1.8 |
| wind gust | 10.81 | 7.24 | 7.08 | 6.83 | 3.5 |
| pressure | 7.82 | 3.82 | 2.83 | 2.26 | 20.1 |

Table 3: Individual sensor changes on weather data with many sensors.

| replaced sensor | Reconstruction Error | | Excess Error | |
|---|---|---|---|---|
| | **ASC** | **ASC**$^{\mathsf{SEL}}$ | **ASC** | **ASC**$^{\mathsf{SEL}}$ |
| temperature (°F) | 0.47 | 0.38 | 0.34 | 0.22 |
| humidity (%) | 0.53 | 0.47 | 0.42 | 0.31 |
| dew point (°F) | 0.47 | 0.44 | 0.37 | 0.25 |
| wind speed (mph) | 5.04 | 4.83 | 4.36 | 3.71 |
| wind gust (mph) | 6.28 | 5.61 | 4.75 | 3.96 |
| pressure (Pa) | 3.17 | 1.68 | 2.68 | 1.04 |

changes, where **ASC**$^{\mathsf{SEL}}$ uses $N_{\mathsf{ref}} = N_{\mathsf{new}} = 10$. In terms of reconstruction errors, **ASC**$^{\mathsf{SEL}}$ improves over **ASC** in all cases. Note that **ASC**$^{\mathsf{SEL}}$ outperforms **ASC** in Table 1, which reveals that a large pool of reference and new sensors actually help. In contrast, **ASC** performs worse than itself in Table 1 due to overfitting. This demonstrates the efficacy of our sensor selection procedure when the number of sensors is large. In terms of excess errors, **ASC**$^{\mathsf{SEL}}$ achieves smaller values than **ASC**, consistent with the fact that **ASC**$^{\mathsf{SEL}}$ learns better reconstruction functions. The excess errors on wind speed and wind gust are relatively large, because these sensor values exhibit large variances and are difficult to reconstruct. We observe similar trends in the scenario of compound sensor changes.

## Results on UUV Data

We gather UUV data by letting a UUV travel from a starting point to an end point in a simulated environment. The UUV contains propeller RPM sensor, waterspeed sensor and a compound sensor called Doppler Velocity Log (DVL) sensor. The DVL sensor consists of seven individual sensors including surge, heave, sway, pitch, roll, depth, and heading. Each sensor produces a sensor value every second. We simulate 20 trips[7] and collect all sensor values. The concatenated sensor values in 10 trips are used as the source domain, and the remaining are used as the target domain. We examine reconstruction errors on surge (m/s), heave (m/s) and sway (m/s) whose sensor values are crucial for higher-layer software. To simulate new sensors, we use a biased version for the surge, heave and sway sensors. The biased version offsets the original sensor values by a sensor-specific bias. We set the bias to $3\sigma$, where $\sigma$ is the standard deviation of the original sensor values.

[7]The trajectory of the UUV varies each time due to different starting/end points and water current.

**Individual sensor changes**. We treat each of the surge, heave and sway sensors as the replaced sensor, and the remaining sensors as reference sensors. Table 4 compares reconstruction errors of different methods, where **ASC** improves over the best baseline by an average of 8.8%. The improvement on sway is less than that on surge and heave because the correlations between sway and reference sensors are smaller. **Refer** and **Refer-Z** always outperform **Replace**, consistent with our observations on weather data.

Table 4: Reconstruction errors (RMSE) on UUV data for individual sensor changes.

| Sensor | **Replace** | **Refer** | **Refer-Z** | **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 |

**Compound sensor changes**. We treat all sensors in the DVL compound sensor as the replaced sensors, and the propeller RPM and waterspeed sensor as reference sensors. Table 5 reports the results. **ASC** performs better than **Refer** and **Refer-Z** on surge and heave, but performs negligibly worse on sway. Compared to Table 4, the improvement decreases for each sensor because fewer reference sensors are used.

Table 5: Reconstruction errors (RMSE) on UUV data for compound sensor changes.

| Sensor | **Replace** | **Refer** | **Refer-Z** | **ASC** | Imp.(%) |
|---|---|---|---|---|---|
| surge | 2.47 | 0.71 | 0.67 | 0.62 | 6.0 |
| heave | 0.094 | 0.026 | 0.026 | 0.024 | 3.4 |
| sway | 2.31 | 0.78 | 0.75 | 0.75 | -0.5 |

## Conclusion

We addressed the novel problem of automatically adapting to sensor changes in the course of building long-lived, survivable software. We presented a machine learning approach capable of exploiting new sensors, scaling to many sensors as well as estimating the adaptation quality. Supported by our empirical study in two important practical domains, i.e., the weather and UUV domains, the proposed approach outperforms baseline methods that do not effectively leverage new sensors. As future work, we would like to incorporate our approach into survivable software systems that rely on a large number of different kinds of sensors.

## Acknowledgements

This material is based upon work supported by the United States Air Force and the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-16-C-0045. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force and DARPA.

## References

Alduchov, O. A., and Eskridge, R. E. 1996. Improved magnus form approximation of saturation vapor pressure. *Journal of Applied Meteorology* 35(4):601–609.

Aminikhanghahi, S., and Cook, D. J. 2016. A survey of methods for time series change point detection. *Knowledge and Information Systems* 1–29.

Basseville, M.; Nikiforov, I. V.; et al. 1993. *Detection of abrupt changes: theory and application*, volume 104. Prentice Hall Englewood Cliffs.

Brodsky, E., and Darkhovsky, B. S. 2013. *Nonparametric methods in change point problems*, volume 243. Springer Science & Business Media.

Dereszynski, E. W., and Dietterich, T. G. 2011. Spatiotemporal models for data-anomaly detection in dynamic environmental monitoring campaigns. *ACM Transactions on Sensor Networks (TOSN)* 8(1):3.

Dereszynski, E. W., and Dietterich, T. G. 2012. Probabilistic models for anomaly detection in remote sensor data streams. *arXiv preprint arXiv:1206.5250.*

Dietterich, T. G.; Dereszynski, E. W.; Hutchinson, R. A.; and Sheldon, D. R. 2012. Machine learning for computational sustainability. In *IGCC*, 1.

Elnahrawy, E., and Nath, B. 2004. Context-aware sensors. In *European Workshop on Wireless Sensor Networks*, 77–93. Springer.

Friedman, J.; Hastie, T.; and Tibshirani, R. 2001. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin.

Gubbi, J.; Buyya, R.; Marusic, S.; and Palaniswami, M. 2013. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems* 29(7):1645–1660.

Gustafsson, F., and Gustafsson, F. 2000. *Adaptive filtering and change detection*, volume 1. Citeseer.

Hughes, J.; Sparks, C.; Stoughton, A.; Parikh, R.; Reuther, A.; and Jagannathan, S. 2016. Building resource adaptive software systems (brass): Objectives and system evaluation. *ACM SIGSOFT Software Engineering Notes* 41(1):1–2.

Lai, T. T.-T.; Chen, W.-J.; Li, K.-H.; Huang, P.; and Chu, H.-H. 2012. Triopusnet: Automating wireless sensor network deployment and replacement in pipeline monitoring. In *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, 61–72. ACM.

Pan, S. J., and Yang, Q. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22(10):1345–1359.

Pimentel, M. A.; Clifton, D. A.; Clifton, L.; and Tarassenko, L. 2014. A review of novelty detection. *Signal Processing* 99:215–249.

Shi, Y., and Knoblock, C. 2017. Learning with previously unseen features. *IJCAI.*

Tong, B.; Wang, G.; Zhang, W.; and Wang, C. 2011. Node reclamation and replacement for long-lived sensor networks. *IEEE Transactions on Parallel and Distributed Systems* 22(9):1550–1563.

Zhang, Y.; Meratnia, N.; and Havinga, P. 2010. Outlier detection techniques for wireless sensor networks: A survey. *IEEE Communications Surveys & Tutorials* 12(2):159–170.