To Compute or Not to Compute? Adaptive Smart Sensing in Resource-Constrained Edge Computing

Luca Ballotta ⦿ , *Member, IEEE*, Giovanni Peserico ⦿ , Francesco Zanini ⦿ , and Paolo Dini ⦿

Abstract—We consider a network of smart sensors for an edge computing application that sample a time-varying signal and send updates to a base station for remote global monitoring. Sensors are equipped with sensing and compute, and can either send raw data or process them on-board before transmission. Limited hardware resources at the edge generate a fundamental *latency-accuracy trade-off:* raw measurements are inaccurate but timely, whereas *accurate* processed updates are available after *processing delay*. Hence, one needs to decide when sensors should transmit raw measurements or rely on local processing to maximize network monitoring performance. To tackle this sensing design problem, we model an estimation-theoretic optimization framework that embeds both computation and communication latency, and propose a Reinforcement Learning-based approach that dynamically allocates computational resources at each sensor. Effectiveness of our proposed approach is validated through numerical experiments motivated by smart sensing for the Internet of Drones and self-driving vehicles. In particular, we show that, under constrained computation at the base station, monitoring performance can be further improved by an *online sensor selection*.

Index Terms—Communication latency, computation latency, edge computing, Q-learning, resource allocation, sensing design.

I. INTRODUCTION

DISTRIBUTED computation scenarios such as the Internet of Things and Industry 4.0 represent a major breakthrough in engineering applications, whereby coordination of sensing and actuation moves away from classical centralized controllers to servers and devices at the network edge. This empowers multiple local systems to achieve together complex goals at global level: this happens with management of electricity and energy harvesting in smart grids [1], [2], resource utilization in smart agriculture [3], [4], modularization and productivity enhancement in Industry 4.0 [5], [6], [7], urban traffic with interconnected vehicles [8], [9], and space-air-ground services [10].

In particular, recent advances in both embedded electronics, with powerful micro controllers and GPU processors [11], [12], and new-generation communication protocols for massive networks, such as 5G [13], [14], are currently pushing network systems to rely on sensors and, more in general, edge devices to carry most of the computational burden. Indeed, distributed computation paradigms such as edge and fog computing [15], [16], [17], [18], [19] and federated and decentralized learning [20], [21], [22], even though still in their infancy, enjoy febrile activity and excitement across the research community.

Despite the growing resources and technological development, emerging edge technologies are still limited compared to centralized servers: indeed, edge devices are forced to trade several factors, such as hardware cost for processing speed for energy consumption. In particular, data processing on devices at the edge requires a non-negligible computational time.

In this work, we consider a group of edge smart sensors, such as compute-equipped IoT nodes or UAVs, that measure a signal of interest – e.g., voltage in a smart grid, or movements of vehicles for surveillance – and transmit the measurements to a base station that performs remote global monitoring and possibly decision-making. Limited hardware resources induce a *latency-accuracy trade-off* at each sensor, that can supply either raw, inaccurate samples of the monitored signal or refine those same data on-board by running suitable algorithms, which produce high-quality measurements at the cost of *processing delay* caused by constrained hardware. Such *local processing* may consist of averaging or filtering a batch of noisy samples, or feature extraction from images or other high-dimensional data [23], [24], to mention a few examples. Because the monitored system evolves dynamically, delays in transmitted measurements may hinder usefulness of these in real-time tasks, so that sensing design for multiple, heterogeneous sensors becomes challenging. In particular, as sensors cooperate, it is unclear which of them should rely on local computation to transmit accurate information, and which ones would be better off sending raw data. Also, channel constraints such as limited bandwidth may introduce non-negligible *communication latency*, further increasing complexity of the sensing design. Specifically, local

processing might compress acquired samples, so that transmission of raw data to the base station takes longer.

A. Related Literature

Resource allocation in terms of sensor and actuator selection represents a major research topic in IT, robotics, and control. Classically, the need for selection emerges from maximization of a performance metric subject to limited resource budget, being it of economical, functional (e.g., weight of autonomous platforms), spatial (e.g., locations to place sensors), or other nature. Typical works in this field [25], [26], [27], [28], [29], [30], [31], [32], [33] focus on such budget-related constraints and pay little attention to impact on system dynamics. For example, [25] proposes selection strategies based on coverage probability and energy consumption for a target tracking problem, [33] studies a clustering-based selection to address communication constraints in underwater environments, and [31] tackles placement of cheap and expensive sensors to optimize reconstruction of dynamical variables. The aforementioned works, even though address computation and/or communication issues, either care about energy consumption or address latency in a qualitative way, but do not use that information to compute an exact performance metric that depends on the system dynamics. Another, more control theoretic, body of work exploits tools from set-valued optimization, e.g., submodular functions with matroid constraints [34], or studies analytical bounds [35] or convex formulations [30], [36], yet within a static framework that does not address changes in the overall dynamics.

In a similar realm, control theory is traditionally concerned with either channel-aware estimation and control, or co-design of communication and controller, addressing wireless channel issues such as unreliability, latency, and more in general limited information [37], [38], [39], [40], [41], [42]. For example, [37], [42] are concerned with rate-constrained stabilizability, while [38], [41] address LQR and LQG control. More recently, performance of wireless cyber-physical systems subject to state and input constraints has been thoroughly investigated leveraging model-based prediction and optimization tools such as MPC [43], [44], [45], [46]. However, also this line of work does not consider processing-dependent delays and their effect on dynamics and performance. Even in recent work on sensing, communication, and control co-design [47], [48] there is no unifying framework that exactly relates sensing and computation on resource-limited platforms to estimation and control performance in dynamical networks. A novel framework concerned with an adaptive design for LQG control which addresses accuracy-dependent sensing latency is presented in [49]. However, it considers a single sensor and proposes a heuristic solution with no theoretical guarantees.

A recent body of literature tailored to edge and fog computing studies distributed computation on resource-constrained devices, focusing on minimization of delays [50], [51], [52], [53] or latency-dependent energy consumption [54]. While there is a clear, empirically supported intuition that outdated sensory information is detrimental to performance through the dynamical nature of monitored systems, the above works do not address

the true performance metrics (which may be unknown or too complicated to compute), but employ heuristic proxies (e.g., delays) without quantifying impact on closed-loop performance.

Finally, a similar trend is found within a recent body of the communications literature on Age of Information (AoI) [55], [56], [57], [58], a metric that quantifies the time elapsed since the latest received update from a source of information. These works focus on minimizing quantities related to AoI of updates, but typically neglect dynamics of the measured variables. Also, most works, e.g., [59], [60], assume that the dynamical systems measured by different sensors are uncoupled, limiting applicability of this approach in networked control systems.

B. Novel Contribution and Organization of the Article

In contrast to previous work, we jointly address sensor local processing, computation and communication latency, and system dynamics towards a dynamical smart sensing design.

In [61], the authors proposed a general model for a processing network, including impact of computation-dependent delays on monitoring performance, and provided a heuristic sensing design. However, that design is static. i.e., sensors cannot adapt to the monitored system during operation, which may hinder performance. For example, time-varying systems generally prevent the optimal sensing configuration to be static. Also, sensors could store incoming samples into an unlimited buffer. We advance such issues through a novel design framework that builds on the insights in [61]. Moreover, this article considerably expands the preliminary version [62] as described next.

First, in Section II-A we propose a novel model for a *processing network* tailored to data acquisition and transmission by resource-constrained smart sensors. These can adapt their local computation overtime and exploit the latency-accuracy trade-off online to maximize global network performance, by choosing to either transmit raw samples or refine data on-board. In addition, motivated by [61], we let sensors temporary stand-by (*sleep*) to alleviate the computational burden for sensor fusion. Roughly speaking, such *online sensor selection* can crucially improve global monitoring performance if the processing resources available at the base station cannot handle large amounts of sensory data in real time. Remarkably, this result goes against the common wisdom that deploying more sensing resources always improve performance.

In Section II-B, we formulate an optimal design problem to manage sensing resources in a network of smart sensors. We do this by computing an estimation-theoretic performance metric that embeds both dynamical parameters and accuracy and delays associated with sensory data. To partially overcome intractability of the problem, in Section III we formulate a simplified version of it, which is tackled via a Reinforcement Learning approach in Section IV, see Fig. 1. Reinforcement Learning, and data-driven methods in general, are now popular in network systems and edge computing because of challenges raised by real-world scenarios [63], [64], [65].

Finally, in Section V we validate our approach with numerical experiments motivated by sensing for autonomous driving and Internet-of-Drone tracking. We address realistic
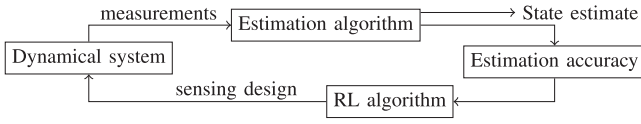
Fig. 1. Scheme of the proposed methodological framework: the RL algorithm learns a sensing design to maximize performance of the estimation algorithm.

communication through an industrial-oriented simulator (OM-NeT++) that accurately models the lower layers of the protocol stack. We show that accounting for latency due to resource constraints can improve performance through a careful allocation of sensing and computation. In particular, the online sensor selection becomes crucial when a large number of sensors is available.

II. SETUP AND PROBLEM FORMULATION

In this section, we first model a processing network composed of smart sensors (Section II-A), and then formulate the sensing design as an optimal estimation problem (Section II-B).

A. System Model

Dynamical System: The signal of interest is described by a time-varying discrete-time linear dynamical system,

$$x_{k+1} = A_k x_k + w_k, \tag{1}$$

where $x_k \in \mathbb{R}^n$ collects the variables (*state*) of the system, $A_k \in \mathbb{R}^{n \times n}$ is the state matrix, and white noise $w_k \sim \mathcal{N}(0, W_k)$ captures model uncertainty. Such class of models is widely used in control applications, by virtue of their simplicity but also powerful expressiveness [28], [66], [67], [68], [69]. For example, a standard approach in control of systems modeled through nonlinear differential equations is to approximate the original model as a parameter- or time-varying linear system, for which efficient control techniques are known [28], [70], [71].

In view of transmission of sensor samples, we assume discrete-time dynamics with time step T, where subscript $k \in \mathbb{N}$ means the kth time instant kT. Without loss of generality, we fix the first instant $k_0 = 1$. The sampling time T represents a suitable time scale for the global monitoring and, possibly, decision-making task at hand. For example, typical values of T are one or two seconds for trajectory planning of ground robots, while higher frequency is required for drones performing a fast pursuit or for self-driving applications.

Smart Sensors: The system modeled by (1) is measured by N smart sensors (or simply sensors) gathered in the set $\mathcal{V} \doteq \{1, \ldots, N\}$, which output a noisy version of the state x_k,

$$y_k^{(i)} = x_k + v_k^{(i)}, \qquad v_k^{(i)} \sim \mathcal{N}(0, V_{i,k}), \tag{2}$$

where $y_k^{(i)}$ is the measurement produced by the ith sensor at time k, for any $i \in \mathcal{V}$, and $v_k^{(i)}$ is measurement noise.

Smart sensors are equipped with processing capabilities alongside standard sensing hardware, and can either transmit raw samples of the signal x_k or locally process acquired samples to provide refined measurements. For example, a smart camera may send raw frames or run computer vision algorithms on the acquired images to get high-quality information, as in typical robot navigation applications where informative features are extracted from visual data. Symbol $y_k^{(i)}$ indifferently refers to raw or processed measurements: as formalized in Assumption 1, the difference between such two kinds of data is embedded into the measurement noise covariance $V_{i,k}$.

Remark 1 (Sensor processing): We consider the case where sensor local processing is *static*, that is, sensors can refine the current sample (as in [49], [61]), but do not (re-)process past samples. This model is suitable to devices that provide data without need (or possibility) of tracking the history of the measured signal, which is handled by the base station. This is different than, e.g., works [72], [73], where sensor processing is adaptive and involves the history of collected samples. Although it might be possible to integrate such kind of processing into our framework, this is a compelling research direction that will be explored in the future.

Sensors face a *latency-accuracy trade-off* through limited hardware: raw data are less accurate, but local data processing introduces extra computational delays that make refined updates more outdated with respect to the current state of the system.

For example, consider a car that is approximately moving at constant speed, with w_k capturing small unmodeled accelerations: as the car moves, knowledge of its real-time position through the nominal model (constant speed) becomes more and more imprecise because of unknown accelerations hidden in w_k, which make the car drift away from its nominal trajectory. In this case, a sensor may prefer to sample the system (e.g., collect positions of the car) more often, rather than spending time to obtain precise, but outdated, position measurements.

We formally model the latency-accuracy trade-off with the next assumptions. We also introduce a third operating mode (*sleep mode*) that lets sensors stand-by. The usefulness of sleep mode is associated with limited computational resources for aggregation of sensory data, and will be motivated in the paragraph "Base Station" below and in Section II-B.

Assumption 1 (Sensing modes): Each sensor $i \in \mathcal{V}$ can be in *raw*, *processing*, or *sleep* mode.

Raw mode: measurements are generated after delay $\tau_{i,\text{raw}}$ with noise covariance $V_{i,k} \equiv V_{i,\text{raw}}$.

Processing mode: measurements are generated after *processing delay* $\tau_{i,\text{proc}}$ with noise covariance $V_{i,k} \equiv V_{i,\text{proc}}$.

Sleep mode: the sensor is temporary set idle (*asleep*): neither data sampling nor transmission occur under this mode.

Assumption 2 (Latency-accuracy trade-off): For each sensor $i \in \mathcal{V}$, it holds $\tau_{i,\text{proc}} > \tau_{i,\text{raw}}$ and $V_{i,\text{raw}} \succ V_{i,\text{proc}}$.[1]

Similarly to [61], Assumption 2 models high accuracy through long computation (Fig. 2) and "small" covariance (intensity) of measurement noise, e.g., raw distance measurements with uncertainty of 1m^2 while processed ones of 0.1m^2.

Next, we define how local operations are ruled overtime.

Definition 1 (Sensing policy): A *sensing policy* for the ith sensor is a sequence of categorical decisions $\pi_i \doteq \{\gamma_k^i\}_{k \geq k_0}$. If

---

[1]Even though Löwner order obeyed by covariance matrices is partial, we require it in our model so that the latency-accuracy trade-off is well defined.
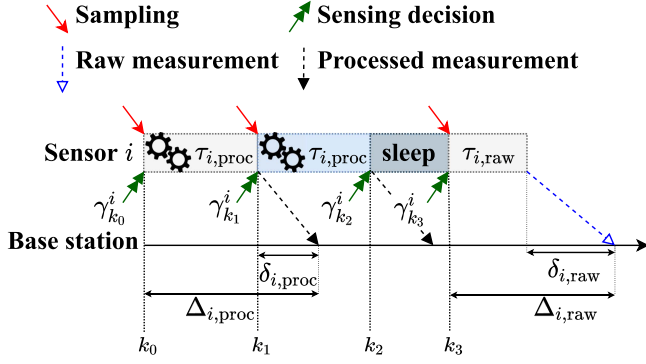
Fig. 2. Data collection and transmission. Computation at the ith sensor is ruled by sensing policy π_i. Here, sensing decisions $\{\gamma^i_{k_j}\}^3_{j=0} = \{\mathrm{p, p, s, r}\}$ are shown and (4) reads $\mathcal{K}_i[0] = s^0_i(k_0) = k_0$, $\mathcal{K}_i[1] = s_i(k_0) = k_1$, and $\mathcal{K}_i[2] = s_i(k_1) = k_3$. Measurements are received after delays induced by local computation (rectangular blocks) and communication (dashed arrows). For example, under $\gamma^i_{k_0} = \mathrm{p}$, the sample acquired at time k_0 is first processed (with processing delay $\tau_{i,\mathrm{proc}}$), then transmitted at time $k_1 = k_0 + \tau_{i,\mathrm{proc}}$ (with communication delay $\delta_{i,\mathrm{proc}}$), and finally received at the base station at time $k_1 + \delta_{i,\mathrm{proc}} = k_0 + \Delta_{i,\mathrm{proc}}$ (with delay at reception $\Delta_{i,\mathrm{proc}}$).

$\gamma^i_k = \mathrm{r}$, measurement $y^{(i)}_k$ is transmitted raw; if $\gamma^i_k = \mathrm{p}$, $y^{(i)}_k$ is processed; if $\gamma^i_k = \mathrm{s}$, no measurement is acquired at time k.

According to Definition 1, different sensing modes can be alternated online. However, because of constrained resources, a sensor cannot acquire measurements arbitrarily often. The actual sampling frequency is determined as formalized next.

Assumption 3 (Sampling frequency): Assume that the ith sensor acquires a sample at time k under either raw ($\gamma^i_k = \mathrm{r}$) or processing ($\gamma^i_k = \mathrm{p}$) mode. Let time k' be defined as

$$k' \doteq \begin{cases} k + \tau_{i,\mathrm{raw}} & \text{if } \gamma^i_k = \mathrm{r} \\ k + \tau_{i,\mathrm{proc}} & \text{if } \gamma^i_k = \mathrm{p} \end{cases}. \tag{3a}$$

Then, the next sample (under any mode) occurs at time $s_i(k)$,

$$s_i(k) \doteq \min_{h \in \mathbb{N}} \{ h \geq k' : \gamma^i_h \neq \mathrm{s} \}. \tag{3b}$$

Finally, the sequence of all sampling instants $\mathcal{K}_i$ is given by

$$\mathcal{K}_i = \left\{ s^l_i(k_0) \right\}_{l \geq 0}, \tag{4a}$$

where consecutive sampling times are defined by the recursion

$$s^{l+1}_i(k) = s_i \left(s^i_i(k) \right)$$
$$s^0_i(k_0) \doteq \min_{h \in \mathbb{N}} \{ h \geq k_0 : \gamma^i_h \neq \mathrm{s} \}, \tag{4b}$$

and $\mathcal{K}_i[l]$ denotes the lth element of the sequence, with $l \in \mathbb{N}$.

In words, Assumption 3 states that sensors can acquire a new sample only after the previous measurement has been transmitted. This is a realistic assumption if agents have limited storage resources [74]. The effect of a sensing policy on sampling and local data processing is illustrated in Fig. 2.

Communication Channel: All sensors transmit data to a common base station through a shared communication channel, which is wireless or wired according to the application requirements. The channel induces *communication latency* that may further delay transmitted updates and depends on several factors such as transmission medium, network traffic, and interference.
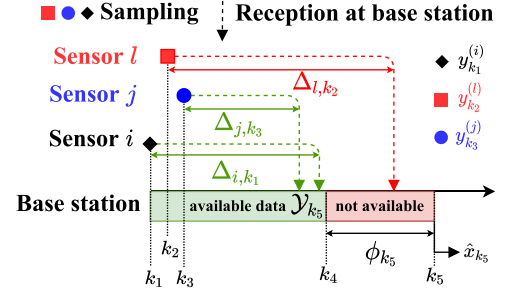

Fig. 3. Data processing at the base station. Resource-constrained centralized processing introduces *fusion delay* ϕ_{k_5} to estimate x_{k_5}. Measurements $y^{(i)}_{k_1}$ and $y^{(j)}_{k_3}$ are received before computation starts at time $k_4 = k_5 - \phi_{k_5}$ and are used to compute $\hat{x}_{k_5}$, i.e., $y^{(i)}_{k_1}, y^{(j)}_{k_3} \in \mathcal{Y}_{k_5}$, while $y^{(l)}_{k_2}$ is received after time k_4 and cannot be used in estimation of x_{k_5}, i.e., $y^{(l)}_{k_2} \notin \mathcal{Y}_{k_5}$.

We let $\delta_{i,\mathrm{raw}}$ and $\delta_{i,\mathrm{proc}}$ denote communication delays of raw and processed data transmitted by the ith sensor, respectively. In general, $\delta_{i,\mathrm{raw}}$ and $\delta_{i,\mathrm{proc}}$ might differ depending on possible data compression due to processing. In case $\delta_{i,\mathrm{raw}} = \delta_{i,\mathrm{proc}}$, we denote both delays by δ_i. The total delay experienced by updates from sampling to reception at the base station is given by $\Delta_{i,\mathrm{raw}} = \tau_{i,\mathrm{raw}} + \delta_{i,\mathrm{raw}}$ for raw and $\Delta_{i,\mathrm{proc}} = \tau_{i,\mathrm{proc}} + \delta_{i,\mathrm{proc}}$ for processed data. Data sampling, processing, and transmission are depicted in Fig. 2.

Base Station: Data are transmitted to a base station in charge of estimating the state of the system x_k in real time. Such estimation enables remote global monitoring and decision-making, e.g., coordinated tracking or exploration. Let $\hat{x}_k$ denote the real-time estimate of x_k. In view of the sequential nature of centralized data processing, the real-time estimate of x_k is computed in ϕ_k time (*fusion delay*), which is proportional to the amount of data used in the update [61]. Consider Fig. 3: from time k_1 through k_4, new data are received at the base station (green dashed arrows). If the estimation routine starts at time k_4, it takes ϕ_{k_5} to process all newly received sensory data (possibly, also old ones if some data arrive out of sequence), and hence the next updated state estimate, $\hat{x}_{k_5}$, will be available at time $k_5 = k_4 + \phi_{k_5}$. Hence, fusion delays induce open-loop predictions that degrade quality of the computed estimates (similarly to what discussed about local sensor processing), and motivate sleep mode to reduce the incoming stream of sensory data and improve overall performance [61].

Assumption 4 (Available sensory data): In view of Assumptions 1, 3, all sensory data available at the base station and used to compute $\hat{x}_k$ at time k are

$$\mathcal{Y}_k \doteq \bigcup_{i \in \mathcal{V}} \bigcup_{l \in \mathbb{N}} \left\{ \left(y^{(i)}_{\mathcal{K}_i[l]}, V_{i, \mathcal{K}_i[l]} \right) : \mathcal{K}_i[l] + \Delta_{i, \mathcal{K}_i[l]} + \phi_k \leq k \right\}$$

$$\Delta_{i, \mathcal{K}_i[l]} \doteq \begin{cases} \Delta_{i,\mathrm{raw}} & \text{if } \gamma^i_{\mathcal{K}_i[l]} = \mathrm{r} \\ \Delta_{i,\mathrm{proc}} & \text{if } \gamma^i_{\mathcal{K}_i[l]} = \mathrm{p} \end{cases}, \tag{5}$$

where the lth measurement from the ith sensor $y^{(i)}_{\mathcal{K}_i[l]}$ is sampled at time $\mathcal{K}_i[l]$ and received after overall delay $\Delta_{i, \mathcal{K}_i[l]}$, and ϕ_k is the time needed to compute $\hat{x}_k$ at the base station.
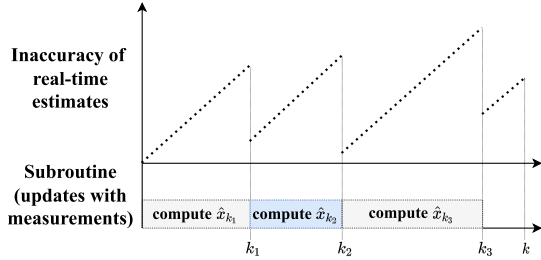
Fig. 4. Real-time estimation at the base station. The state estimate is updated at each point in time (top). Because of limited resources at the base station, open-loop updates are performed whenever fresh sensory data are being processed (bottom), causing estimation to degrade overtime through additive noise w_k in nominal dynamics (1). As soon as the data processing subroutine produces an updated estimate with new measurements, e.g., $\hat{x}_{k_1}$ at time k_1, the estimation inaccuracy is reduced. Note that the top plot is qualitative: the estimate quality does not degrade linearly, in general.

According to Assumption 4, a measurement $y_h^{(i)}$ can be used to compute the estimate of x_k in real time if it is successfully delivered to the base station (with delay at reception $\Delta_{i,h}$) before or at time $k - \phi_k$, where ϕ_k is the amount of time needed to compute $\hat{x}_k$. Data processing at the base station with limited resources and data availability is depicted in Fig. 3.

Remark 2 (Real-time estimation): Based on the above discussion, new data cannot be used by an estimation procedure between times k and $k + \phi_k$. In a real system, a real-time state estimate must always be available for effective monitoring. We assume that two parallel jobs are executed. A support subroutine processes received measurements and computes a state estimate at time k in ϕ_k time (cf. Fig. 3). The real-time estimation routine computes one-step-ahead open-loop updates at each point in time according to the nominal dynamics (1) (progressively degrading estimation quality), and resets when the support subroutine outputs an updated estimate with new measurements (with higher estimate quality).[2] A schematic representation is shown in Fig. 4. Importantly, degradation of estimation in the top plot is not due to lack of new measurements (like in Age of Information literature), but is caused by constrained resources that induce a computational bottleneck in the support subroutine (bottom plot in Fig. 4).

B. Problem Statement

The trade-offs introduced in the previous section call for a challenging sensing design at the network level. In particular, all possible choices of local sensor processing (we address a specific choice for all sensors as a *sensing configuration*) affect global performance in a complex manner, whereby it is unclear which sensors should transmit raw measurements, with poor accuracy and possibly long communication delays, and which ones should refine their samples locally to produce high-quality measurements. In fact, the authors in [61] show that the optimal configuration when considering steady-state performance is nontrivial. Also, the optimal sensing configuration is time varying, in general. Thus, sensing policies $\pi_i, i \in \mathcal{V}$, have to be suitably designed to maximize the overall network performance.

[2]One-step-ahead open-loop steps are assumed computationally cheap.

The state x_k is estimated via Kalman predictor, which is the optimal observer for linear systems with Gaussian disturbances. It can be shown, e.g., via state augmentation, that the Kalman predictor is optimal even with delayed measurements, whereby it suffices to ignore updates associated with missing data (see Appendix A in Supplementary Material). Out-of-sequence arrivals can be handled by recomputing all predictor steps since the latest arrived measurement has been acquired, or by more sophisticated techniques [75], [76].

Let $\tilde{x}_k \doteq x_k - \hat{x}_k$ the estimation error of Kalman predictor at time k, and let $P_k \doteq \mathrm{Var}(\tilde{x}_k)$ its covariance matrix. We formulate the sensing design as an optimal estimation problem.

Problem 1 (Sensing Design for Processing Network): Given system (1)–(2) and Assumptions 1–4, find the optimal sensing policies $\pi_i, i \in \mathcal{V}$, that minimize the time-averaged estimation error variance with horizon K,

$$\underset{\pi_i \in \Pi_i, i \in \mathcal{V}}{\arg\min} \quad \frac{1}{K} \sum_{k=k_0}^{K} \mathrm{Tr}\left(P_k^\pi\right) \tag{6a}$$

$$\text{s.t.} \quad P_k^\pi = f_{\text{Kalman}}\left(\mathcal{Y}_k^\pi\right) \tag{6b}$$

$$P_{k_0}^\pi = P_0, \tag{6c}$$

where the Kalman predictor $f_{\text{Kalman}}(\cdot)$ computes at time k the state estimate $\hat{x}_k^\pi$ and the error covariance matrix P_k^π using data $\mathcal{Y}_k^\pi$ available at the base station according to $\pi \doteq \{\pi_i\}_{i \in \mathcal{V}}$, and Π_i gathers all causal sensing policies of the ith sensor.

Remark 3 (Impact of processing on estimation): Processed measurements are more accurate than raw ones: hence, if delays were neglected, the optimal (trivial) design would be to always process, because this yields the smallest variance of measurement noise (Assumption 1) and minimizes the estimation error variance of the Kalman predictor when updates with measurements are performed. However, computational delays associated with data processing introduce extra open-loop steps that increase the error variance, making the optimal design nontrivial. In other words, uncertainty about the true dynamics (captured in (1) by noise w_k) makes refined measurements be less informative about the *current* state of the system, so that high accuracy alone might not pay off in real-time monitoring.

Remark 4 (Novelty of sensor selection): Sleep mode actually implements an *online sensor selection*, whereby sleeping sensors do not supply data to the base station. We identify two key elements that make our framework fundamentally different from standard sensor selection in the literature. First, while we exploit sleep mode towards *optimal performance*, sensors are typically selected to trade performance for available resources under the conventional belief that more sensors yield better performance. In contrast, selection emerges *naturally* in our framework *to maximize performance* in view of the computational bottleneck at the base station that may increase the estimation cost in (6). Moreover, rather than a *static* selection, we allow for *dynamical switching* to and from sleep mode, which both enables performance improvement through richer design options and is more challenging to optimize.

III. SENSING POLICY: A CENTRALIZED IMPLEMENTATION

Problem 1 is combinatorial in the number of sensors and raises a computational challenge in finding efficient sensing policies, because the search space may easily explode. For example, 10 sensors yield $2^{10} = 1024$ possible sensing configurations at each sampling instant. Also, Problem 1 requires to design a potentially asynchronous schedule for each sensor, which is an additional combinatorial problem in the time horizon K. To further complicate things, a sensing policy π_i not only affects delay and accuracy of measurements supplied by the ith sensor, but also determines the very sequence of sampling instants $\mathcal{K}_i$ (cf. (3)–4), augmenting the search space to all possible time sequences over K steps. In particular, sleep mode represents a computational challenge, because it requires evaluation of all instants subsequent to its activation to decide the best time for triggering a new update.

To partially ease the intractability of the problem, and motivated by practical applications, we restrict the domain of candidate sensing policies to reduce problem complexity while maintaining a meaningful setup. First, we look at the simple but relevant scenario with a homogeneous network and motivate the design of a centralized policy in Section III-A. We then go back to the general scenario with a heterogeneous network and formulate a simplified version of Problem 1 in Section III-B.

A. Homogeneous Network

Sensor Model: In this scenario, all smart sensors have equal measurement noise distributions,

$$y_k^{(i)} = x_k + v_k^{(i)}, \qquad v_k^{(i)} \sim \mathcal{N}(0, V_k), \qquad (7)$$

with $V_k = V_{\text{raw}}$ or $V_k = V_{\text{proc}}$ for raw and processed data, respectively. Also, all sensors feature identical computational and transmission resources, given by delays $\tau_{\text{row}}, \delta_{\text{row}}$ for raw measurements and $\tau_{\text{proc}}, \delta_{\text{proc}}$ for processed measurements, respectively (δ in case of no compression). This *homogeneous network* models the special but relevant case where sensors are interchangeable. This happens for example with sensor networks measuring temperature in plants or chemical concentrations in reactors. Also, this model captures smart sensors collecting high-level environmental information, such as UAVs tracking the position of a body moving in space.

Centralized Policy: In this case, it is sufficient to decide *how many*, rather than *which*, sensors follow a certain mode. Accordingly, we focus on the design of a centralized policy that commands all sensors with no distinctions among them.

Definition 2 (Homogeneous sensing policy): A *homogeneous sensing policy* is a sequence of categorical decisions $\pi_{\text{hom}} = \{\gamma_\ell^{\text{hom}}\}_{\ell=1}^L$. Each decision $\gamma_\ell^{\text{hom}} = (N - n_{\text{s}}, n_{\text{p}})$ is taken at time $k^{(\ell)}$ such that n_{s} sensors are in sleep mode and n_{p} out of the other $N - n_{\text{s}}$ sensors are in processing mode between times $k^{(\ell)}$ and $k^{(\ell+1)}$, with $0 \leq n_{\text{s}} \leq N$ and $0 \leq n_{\text{p}} \leq N - n_{\text{s}}$. Without loss of generality, we set $k^{(1)} = k_0$, $k^{(L)} \leq K$.

In words, the base station decides a configurations for all sensors at predefined time instants, which is both practical for applications and convenient to reduce complexity of the problem. However, decisions may be taken at any times, as long
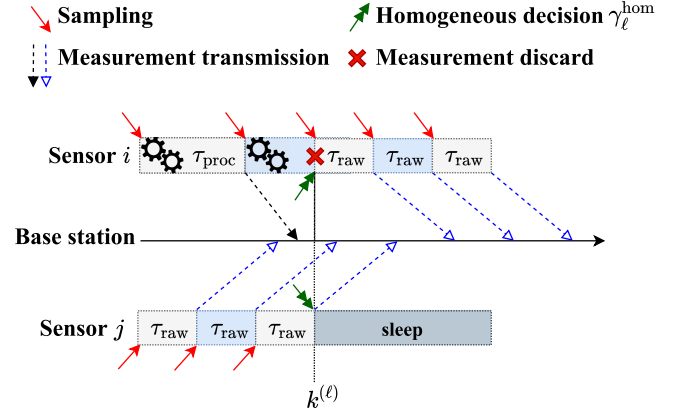

Fig. 5. Homogeneous sensing policy. Sampling and data processing at identical sensors are ruled by policy π_{hom}. Decision γ_ℓ^{hom} is communicated at time $k^{(\ell)}$ and realized at individual sensors as $\gamma_\ell^i = \text{r}$ and $\gamma_\ell^j = \text{s}$. Concurrently, the ith sensor disregards its current processed measurement (red cross) and switches to raw mode, acquiring a new sample at time $k^{(\ell)}$.

as these are consistent with sensor computational delays (e.g., to guarantee that one sample is collected for each decision).

With a slight abuse of notation, to denote the mode of a specific sensor that is following the homogeneous decision γ_ℓ^{hom}, we write $\gamma_\ell^i = \text{m}$ meaning that the ith sensor is set in mode m by the ℓth homogeneous decision, where $\text{m} \in \{\text{r}, \text{p}, \text{s}\}$ can be raw (r), processing (p), or sleep mode (s), respectively. We stress that in this context γ_ℓ^i does not represent a decision of a *single-sensor* sensing policy π_i (as in Definition 1), but all decisions are centralized and γ_ℓ^i denotes the mode that *the base station commands* to the ith sensor through decision γ_ℓ^{hom}.

By design, centralized decisions are communicated regardless of current sensing status. In light of common practice in real-time control [77], [78], [79], [80], [81], we assume what follows.

Assumption 5 (Sampling frequency with homogeneous sensing policy): Decision γ_ℓ^{hom} switches mode of the minimum amount of sensors possible. If the ith sensor switches mode, the measurement currently being acquired or processed (if any) is immediately discarded. If the new commanded mode is either raw or processing, a new sample is acquired according to such new mode right after the decision γ_ℓ^{hom} is communicated.

Formally, given measurement $y_k^{(i)}$ sampled at time $k < k^{(\ell)}$ obeying decision $\gamma_{\ell-1}^{\text{hom}}$, the sampling dynamics (3b) becomes

$$s_i^{\text{hom}}(k) \doteq \begin{cases} k' & \text{if } k^{(\ell)} \geq k' \\ k^{(\bar{\ell})} & \text{otherwise,} \end{cases} \qquad (8a)$$

$$\bar{\ell} \doteq \min_{\ell' \in \{1,\ldots,L\}} \left\{ \ell' : \ell' \geq \ell \wedge \gamma_{\ell'}^i \neq \text{s} \right\}. \qquad (8b)$$

Further, $y_k^{(i)}$ is discarded (not transmitted) if $s_i^{\text{hom}}(k) \neq k'$.

The new sampling mechanism is depicted in Fig. 5. According to Assumption 5, a measurement is not transmitted to the base station if it is not ready when a concurrent decision is communicated. In Fig. 5 the ith sensor discards a measurement whose processing is not completed at time $k^{(\ell)}$, when a new decision switches its mode. Formally, a sensor disregards raw (resp. processed) measurements sampled at time $\bar{k} < k^{(\ell)}$ such that

$\bar{k} + \tau_{\text{row}} > k^{(\ell)}$ ($\bar{k} + \tau_{\text{proc}} > k^{(\ell)}$), i.e., their acquisition ends after a *different* mode is imposed by decision γ_ℓ^{hom} (cf. (8a)). We denote by $\mathcal{Y}_k^{\pi_{\text{hom}}}$ all available data at the base station at time k according to (8) and such discard mechanism imposed by policy π_{hom}, that excludes some data included in $\mathcal{Y}_k$ (cf. (5)).

B. Heterogeneous Network

We now return to the original model (2) with heterogeneous sensors. Without loss of generality, we assume that the sensor set $\mathcal{V}$ is partitioned into M subsets $\mathcal{V}_1, \ldots, \mathcal{V}_M$, where subset $\mathcal{V}_m$, $m \in \{1, \ldots, M\}$, is composed of homogeneous sensors of the mth class. From what discussed in the previous section, it is sufficient to specify how many sensors follow a certain mode within each subset $\mathcal{V}_m$. Hence, we narrow down the domain of all possible policies according to the next definition.

Definition 3 (Network sensing policy): A *network sensing policy* is a collection $\pi_{\text{net}} \doteq \{\pi_{\text{hom},m}\}_{m=1}^M$, where each homogeneous sensing policy $\pi_{\text{hom},m}$ is associated with homogeneous sensor subset $\mathcal{V}_m$, and all homogeneous decisions $\{\gamma_\ell^{\text{hom},m}\}_{m=1}^M$ are communicated together at time $k^{(\ell)}$.

In Definition 3, decision times are fixed like in the homogeneous case, so that decisions are communicated to all sensors at once. At time $k^{(\ell)}$, homogeneous decision $\gamma_\ell^{\text{hom},m}$ involves sensors in $\mathcal{V}_m$, and the overall sensing configuration is given by the ensemble of such decisions. All data available at the base station at time k are collected in $\mathcal{Y}_k^{\pi_{\text{net}}} \doteq \{\mathcal{Y}_k^{\pi_{\text{hom},m}}\}_{m=1}^M$.

Finally, we get the following simplified problem formulation.

Problem 2 (Centralized Sensing Design for Processing Network): Given system (1)–(2) with Assumptions 1–5, find the optimal network sensing policy π_{net} that minimizes the time-averaged estimation error variance with horizon K,

$$\underset{\pi_{\text{net}} \in \Pi_{\text{net}}}{\arg\min} \quad \frac{1}{K} \sum_{k=k_0}^K \text{Tr}\left(P_k^{\pi_{\text{net}}}\right) \tag{9a}$$

$$\text{s.t.} \qquad P_k^{\pi_{\text{net}}} = f_{\text{Kalman}}\left(\mathcal{Y}_k^{\pi_{\text{net}}}\right) \tag{9b}$$

$$P_{k_0}^{\pi_{\text{net}}} = P_0, \tag{9c}$$

where the Kalman predictor $f_{\text{Kalman}}(\cdot)$ computes at time k the state estimate $\hat{x}_k^{\pi_{\text{net}}}$ and the error covariance matrix $P_k^{\pi_{\text{net}}}$, using data available at the base station according to π_{net}, and Π_{net} is the space of causal network sensing policies.

IV. REINFORCEMENT LEARNING ALGORITHM

By assuming complete knowledge of delays and measurement noise covariances affecting sensors in the different modes, both Problems 1 and 2 can be analytically solved. However, the computation of the exact minimizer requires to keep track of all starts and stops of data transmissions for each sensor, resulting in a cumbersome procedure which admits no closed-form expression, and requires to solve a combinatorial problem which does not scale with the number of sensors. Moreover, the assumptions considered in the formulation of the problem may be too conservative in real-life scenarios, and the latter method cannot be relaxed. Indeed, as long as either delays or covariances are not explicitly known or have some variability, i.e., they can be
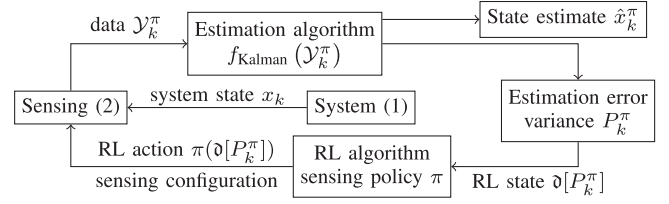

Fig. 6. Learning framework. The RL algorithm receives accuracy of estimates (state) and outputs sensing configurations (action) that affect sensory data.

modeled by proper random variables, the minimization becomes intractable. This is true even if the expectations of these random variables are known, since the dynamics in Problem 2 lead to a nonlinear behavior for the quantity to be minimized.

For the reasons above, we tackle the problem of choosing the optimal sensing policy to minimize estimation uncertainty through a Reinforcement Learning (RL) algorithm, which executes a sequential decision-making suitable for a dynamical sensing design and can flexibly address the general problem formulation. Specifically, the RL algorithm is run at the base station and implements a network sensing policy π_{net} by iteratively choosing a sensing configuration at each time $k^{(\ell)}$. A scheme of the overall framework is given in Fig. 6.

A. Optimizing Latency-Accuracy Trade-Off

The Reinforcement Learning problem of maximizing a reward function through the correct sequence of actions is addressed in this work by the Q-learning algorithm. The latter is a *model-free* and *off-policy* algorithm which updates the current estimate of the action-value-function targeting an *optimistic* variant of the temporal-difference error. In a finite Markov Decision Process, this approach converges to the *optimal* action-value function under standard Monro-Robbins conditions [82], and is efficient with respect to standard competitors [83], [84].

With regard to Problem 1, policy π_i is composed of categorical variables corresponding to sensing modes, and characterizes the potential for intervention in the operations of the ith sensor. The constraints due to the centralized implementation in Problem 2 allow us to consider a single policy $\pi_{\text{net}} : \mathcal{S} \to \mathcal{A}$ describing how many sensors are required to process or sleep for each subset $\mathcal{V}_m$. In particular, action $a \in \mathcal{A}$ is described by M pairs of integers specifying, for each group $\mathcal{V}_m$, i) how many sensors transmit and ii) how many out of the latter ones are in processing mode (cf. Definition 2). For example, if $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$ with $|\mathcal{V}_1| = 4$ and $|\mathcal{V}_2| = 5$, action $a = \{(2,1), (3,0)\}$ means that, within $\mathcal{V}_1$, two sensors are commanded to transmit, one of these being in processing mode, and the other two sleep, and similarly for $\mathcal{V}_2$.

Since we aim to minimize the time-averaged error variance (9a), a straightforward metric to be chosen as reward function is the negative trace of matrix $P_k^{\pi_{\text{net}}}$, which evolves according to the Kalman predictor with delayed updates. In the considered framework the base station is allowed to change sensing configuration (corresponding to a new action) at each time $k^{(\ell)}$, therefore a natural way of defining the reward is to take the average of the negative trace of the covariance during the interval between times $k^{(\ell)}$ and $k^{(\ell+1)}$, so that the base

station can appreciate the performance of a particular sensing configuration in that interval.

This leads to the following instantiation of the RL problem,

$$\max_{\pi_{\text{net}} \in \Pi_{\text{net}}} \quad -\mathbb{E}\left[\sum_{\ell=1}^{L} \frac{\gamma^{\ell}}{k^{(\ell+1)} - k^{(\ell)}} \sum_{k=k^{(\ell)}}^{k^{(\ell+1)}} \text{Tr}\left(P_k^{\pi_{\text{net}}}\right)\right] \quad (10a)$$

$$\text{s.t.} \quad P_k^{\pi_{\text{net}}} = f_{\text{Kalman}}\left(\mathcal{Y}_k^{\pi}\right) \quad (10b)$$

$$P_{k_0}^{\pi_{\text{net}}} = P_0, \quad (10c)$$

with $k^{(L+1)} \doteq K$. The quantity of interest is the trace of the error covariance and thus a straightforward approach would take $\mathcal{S} = \mathbb{R}^+$. To keep the Q-learning in a tabular (finite) setting, we discretize the state space through a function $\mathfrak{d} : \mathbb{R}^+ \to \mathbb{N}^+$. In particular, the image of $\mathfrak{d}[\cdot]$ is given by M bins, which were manually tuned in our numerical experiments to yield a fair representation of the values of $P_k^{\pi_{\text{net}}}$ observed along the episodes. Then, based on the bin associated with $\text{Tr}(P_{k^{(\ell)}})$, the agent outputs a sensing configuration $a \in \mathcal{A}$ through $\pi_{\text{net}}(\cdot)$ at each time $k^{(\ell)}$, given by $a_\ell = \pi_{\text{net}}(\mathfrak{d}[\text{Tr}(P_{k^{(\ell)}})])$.

Notably, choosing $\gamma = 1$ and time intervals $[k^{(\ell)}, k^{(\ell+1)}]$ of equal length matches (10a) and the objective cost (9a) exactly.

Remark 5 (System dynamics and computational complexity): The Reinforcement Learning procedure is concerned only with the selection of the sensing scheme and not with computation of the estimate $\hat{x}_k$ (contrary to data-driven estimation). In particular, the sensor configuration is optimized with respect to the evolution (10b) of the covariance matrix P_k induced by the Kalman predictor and not with respect to the actual system dynamics (1), which are measured by the sensors. The Reinforcement Learning step is then independent from the dimension n of the original system (1), because it deals only with the error variance of state estimates, while the estimation is performed by the Kalman predictor in a model-based fashion.

B. Discussion: Challenges of Reinforcement Learning

While the proposed RL-based solution can tackle Problem 2 in more flexible and efficient way than brute-force or greedy search, it also comes with nontrivial limitations due to computational and performance challenges of RL algorithms, which we discuss next. Tackling such challenges requires dedicated efforts that will be addressed in follow-up work. Note however that the framework considered in this article is representative of a broad range of control applications, and the following issues do not constitute a threat in the current setting.

First, while we consider a single processing mode in Assumption 1, a smart sensor may in general choose among several options to refine raw samples. For example, a robot equipped with cameras may run multiple geometric inference algorithms for perception, each trading runtime for accuracy [49], [85]. In general, the sensing policy of each sensor might feature several design options (modes), which in turn imply a larger action space for the Q-learning and might raise a nontrivial computational challenge because the total number of actions does not scale with the number of sensors.

Second, even though we focus on a centralized learning technique, this inevitably leads to poor computational scalability especially with heterogeneous sensors, because the action space is the combination of actions of individual sensors. While it is worth pointing out that many control and robotic applications involve either *identical* or *a few different* sensors, for which a centralized learning approach is feasible, investigating computationally efficient strategies to improve the scalability of the training in general is a relevant research question. One way to tackle this challenge might be Multi-Agent Reinforcement Learning [86], where each agent (here, smart sensor) receives the reward from the environment and autonomously trains its own policy, possibly exchanging information with other agents. In this scenario, each agent chooses only its own actions, so that the total number of actions actually scales with the number of agents and permits a computationally scalable training. Another argument in favor of this scenario is the possibility of training *asynchronous* sensing policies tailored to the general problem formulation (6), which is hardly solvable via centralized learning and might turn especially useful to effectively trigger the sleep mode. However, the price to pay is the reduced or absent coordination among the agents, which can slow down the overall training or even prevent convergence.

Last, although the Q-learning algorithm is one of the most widespread algorithms because of its effectiveness and ease of implementation, the proposed procedure could be improved by refining some aspects of the current set up. One of the most challenging aspects is the handling of the continuous state-space, which has been solved through a simple discretisation. The latter can be seen as the simplest instance of function approximation, so there could be better ways of addressing the specific state-space resulting from the present formulation. It is nonetheless remarkable that satisfactory results can already be achieved with this simple version, proving the flexibility of the Q-learning algorithm. Note indeed that the particular framework of a continuous state-space and a discrete action-space significantly reduces the range of algorithms that can be applied to solve the problem addressed in this work. The Q-learning algorithm proved flexible enough to handle the difficulties of the non-Markovian environments in both settings considered in Section V. To assess its effectiveness, the convergence is numerically studied in Appendix C in the Supplementary Material, together with a discussion on how the sample complexity scales with the number of sensors in the homogeneous setting. While more powerful RL algorithms could give better solutions, an extensive investigation is out of the scope of this work, whose goal is to propose a general methodology for the addressed sensing design problem.

V. NUMERICAL SIMULATIONS

In the previous sections, we have presented an estimation-theoretic framework for optimal sensing design under resource constraints at processing units and communication channel, together with a solution approach based on Reinforcement Learning. We next showcase applicability of our setup through two edge-computing scenarios. This allows us to get insight
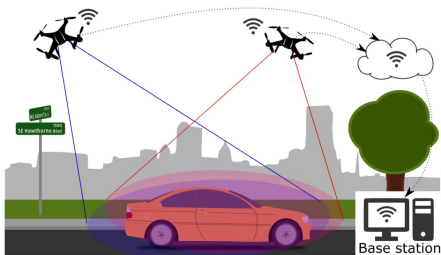
Fig. 7.   Drone tracking simulation setup. The base station estimates the trajectory of the moving target (car) based on visual updates from drones.

TABLE I
SENSOR PARAMETERS FOR DRONE-TRACKING SCENARIO

T	τ_{raw}	τ_{proc}	v_{raw}	v_{proc}
10ms	40ms	140ms	10	1

TABLE II
LEARNING HYPERPARAMETERS FOR DRONE-TRACKING SCENARIO

M	α	ϵ_0	$\epsilon_{\min}$	ϵ_t	γ
5	0.002	0.9	0.1	$\max\left\{\frac{\epsilon_0}{\sqrt{t+1}}, \epsilon_{\min}\right\}$	1

into the structure of optimal sensing, and also shows that our proposed approach can outperform standard design choices.

In Section V, we consider drones for target tracking and see how online sensor selection can improve performance. In Section V-B, we consider smart sensors monitoring an autonomous car to get insight into processing allocation for heterogeneous networks.[3] Finally, in Section V-C we elaborate on the role of Reinforcement Learning in conjunction with a model-based tool such as Kalman filter.

A. Team of Drones for Target Tracking

We simulated a team of 25 drones tracking a vehicle on the road (Fig. 7) modeled as a double integrator [61]. Each drone carries a camera and can either transmit raw frames or run neural object detection on-board, sending fairly precise bounding boxes. We simulated in Python, with parameters in Table I based on experiments in [87], [88] and communication delays $\delta = 10$ ms. We set fusion delays ϕ_k proportional to the number of data that are processed by Kalman predictor to compute $\hat{x}_k$. We addressed an optimization horizon $[0, K]$ split into ten 500 ms-long windows and Q-learning hyperparameters reported in Table II, where t means the tth episode (one episode being one horizon), training for 500000 episodes.

The sensing design policy learned for the horizon is shown in Table III (second row). Notably, only raw mode is chosen: in particular, 10 drones are active in raw mode during the first window and 20 through the rest of the horizon. This means that, with the parameters in Table I, data processing at the edge is inconvenient because the resource constraints of drones induce long processing delays. In addition, the use of sleep mode, that

---

TABLE III
NETWORK SENSING POLICY π_{NET} LEARNED BY Q-LEARNING

v_{proc}	Time window	1	2-10
1	Transmitting (processing) sensors	10 (0)	20 (0)
0.5	Transmitting (processing) sensors	9 (7)	11 (11)
0.1	Transmitting (processing) sensors	8 (7)	14 (14)

The numbers show how many sensors transmit (process) across time.
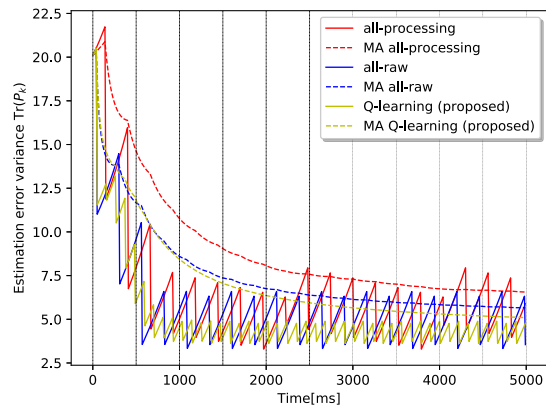
Fig. 8.   Estimation error variance in drone tracking simulation.

TABLE IV
MEAN ERROR VARIANCE IN DRONE-TRACKING SIMULATION

Q-learning (proposed)	All-raw	All-processing
5.10	5.69	6.58

actually implements an *online sensors selection*, improves performance: in words, transmission of data from all drones cannot be efficiently handled by the base station and introduces extra computation latency, with consequent performance degradation. This finding is remarkable because it clashes against the typical assumption that performance improves monotonically with the number of sensors.

We compare our approach against two standard, static design choices: all sensors transmit raw data at all times (*all-raw*) and all sensors refine measurements at all times (*all-processing*). The comparison is shown in Fig. 8 and Table IV. Both baselines are outperformed by optimization (9). Interestingly, our solution also keeps small the Moving Average (MA) of the error variance with respect to the other two designs (see Fig. 8). The largest improvement is recorded at steady-state, while during the transient all curves are very close, with *all-raw* performing best at times. This may have two causes: the transient phase is more difficult to explore for the Q-learning, but also, that seemingly sub-optimal behavior during the first two windows might be necessary given that the learning procedure targets the whole horizon. Indeed, the optimal solution to (9) need not patch together policies that optimize different time windows.

To further investigate the structure of optimal sensing design, we have trained policies with different values of sensor parameters. In particular, we have considered data processing with progressively higher accuracy, quantified by measurement noise

---

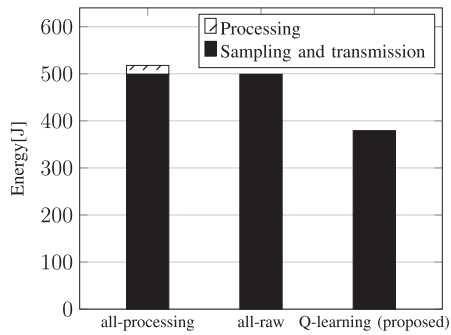[3]Code available at github.com/lucaballotta/ProcessingNetworks-RL.

Fig. 9. Total energy consumption in drone-tracking simulation.

TABLE V
ENERGY CONSUMPTION BREAKDOWN DURING TRANSIENT (TRANS., WINDOWS 1-2) AND AT STEADY STATE (SS., WINDOWS 3-10) FOR SAMPLING AND TRANSMISSION (TX) AND PROCESSING (PROC.) IN DRONE-TRACKING SIMULATION

	Q-learning (proposed)		All-raw		All-processing	
	Tx	Proc.	Tx	Proc.	Tx	Proc.
Trans.	**59.6**J	0	99.8J	0	99.8J	3.8J
Ss.	**319.2**J	0	399.0J	0	399.0J	15.2J
Total	**378.8**J		498.8J		517.8J	

variances $v_{proc} \in \{1, 0.5, 0.1\}$. The learned sensing policies are shown in Table III. As the accuracy of data processing improves, fewer sensors are needed to achieve high estimation quality (small error variance), while the enhanced processing induces to set more sensors to processing mode. This is indeed consistent with intuition and may help in design of real applications. Detailed results for the two additional cases are given in Appendix B in Supplementary Material.

Remark 6 (Energy saving): An appealing side effect of our proposed design through the online sensor selection it induces is reduced energy consumption, which can increase the lifespan of the system. Considering industrial devices such as Genie Nano cameras [89], with typical power consumption of 3.99 W for sampling and transmission, and assuming 0.15 W for data processing [90], the energy consumption under the confronted sensing policies is shown in Fig. 9 and Table V . In particular, our policy uses only 76% of the energy consumed by *all-raw*.

Remark 7 (Computational scalability): As mentioned in Section IV-B, our centralized learning approach can handle small-to-medium network sizes but may struggle when the number of sensors is large. To evaluate how the learning complexity scales with the network, we have run experiments with various numbers of drones, which are reported in Appendix C.

B. Smart Sensing for Self-Driving Vehicle

In our second experiment, we considered a self-driving car traveling at approximately constant speed. Specifically, we considered its transversal position with respect to the center of the lane, which is estimated by an internal controller (base station) that receives data from sensors on-board the car and tracks the car trajectory (Fig. 10), for example to control a lane shift at sustained speed (e.g., for passing or on a highway). The car
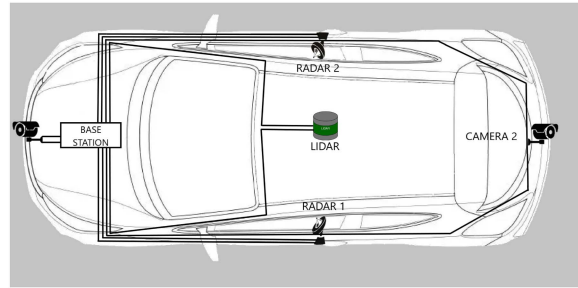
Fig. 10. Autonomous-driving simulation setup. Sensors on-board measure the position of the car and a centralized microcontroller tracks its trajectory.

TABLE VI
SENSOR PARAMETERS FOR AUTONOMOUS DRIVING SCENARIO

	Freq.	τ_{raw}	τ_{proc}	v_{raw}	v_{proc}
Radar	50Hz	20ms	30ms	0.45	0.40
Camera	25Hz	40ms	100ms	0.30	0.05
Lidar	10Hz	100ms	140ms	0.10	0.05

dynamics are modeled through a double integrator, which is a flexible choice used for uncertain dynamics with direct control of accelerations [47], [91], [92], [93], [94]. Given such a model, Kalman predictor is an effective estimator assuming that lateral movements are limited compared to the car speed.

We considered two radar devices, two cameras and one lidar, which are commonly employed in self-driving applications [95]. Many techniques used in autonomous driving exploit lidar point clouds, such as segmentation, detection and classification tasks [96]. Also, radars are emerging as a key technology for such systems. Some of today's self-driving cars, e.g., Zoox, are equipped with more than 10 radars providing 360° surrounding sensing capability under any weather conditions [97]. Finally, camera images are essential to enable commercialization of self-driving cars with autonomy at level 3 [98]. The sensor parameters (Table VI, with $= v_{raw}I$ and $V_{proc} = v_{proc}I$) were chosen based on real-world experiments [99], with sampling period $T = 1$ ms to ensure real-time vehicle control.

The sensor network is designed according to the architecture proposed in [100]: here, smart sensors embed a sensor (e.g., camera) and a microcontroller that can refine raw sensory data to decrease the computational effort for sensor fusion. The base station is a controller inside the car that manages all jobs needed for autonomous driving. Because the application is safety-critical, transmissions occur through two redundant high-speed Ethernet cables. In light of the small number of sensors and transmission speed, we assume that communication latency is negligible with respect to sampling and processing.

Communication was simulated through the discrete-event simulator Objective Modular Network Testbed in C++ (OMNeT++) [101]. This is widely adopted to simulate networks, because it combines standard communication protocols (e.g., IEEE 802.3) and the possibility to create customized procedures exploiting existing modules. Further, it enables realistic simulations by accurately modeling both the electromagnetic environment and the lower layers of the protocol stack (from

TABLE VII
LEARNING HYPERPARAMETERS FOR AUTONOMOUS-DRIVING SCENARIO

M	α	ϵ_0	$\epsilon_{\min}$	ϵ_t	γ
5	0.1	0.2	0.01	$\max\left\{\frac{\epsilon_0}{\sqrt{t+1}}, \epsilon_{\min}\right\}$	1

TABLE VIII
Q-LEARNING POLICY FOR AUTONOMOUS-DRIVING SCENARIO

Fig. 11. Estimation error variance in autonomous-driving simulation.

physical to transportation layers). In our simulations, sensors carry IEEE 802.3 (so-called Ethernet) communication boards.

For training, we considered a time horizon $[0, K]$ split into five time windows with length 300 ms each, and trained for 100000 episodes with hyperparameters reported in Table VII.
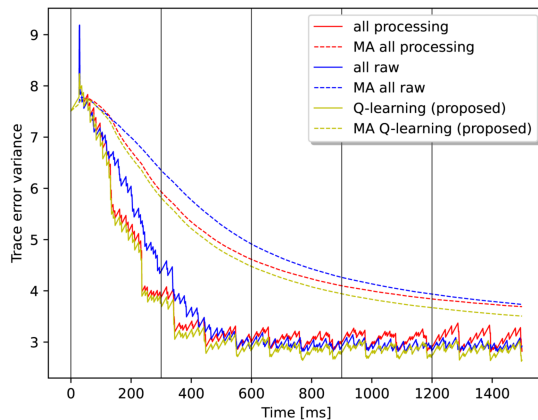
From Table VIII we can infer that the learned policy requires processing almost from all the sensors when the error variance is high (top row). However, the need for processing diminishes with the variance, turning to raw mode both lidar and radars at the smallest values (bottom row). Interestingly, processing mode is always chosen for cameras, revealing that refining of image frames overhangs the additional computational delay. Note that in this case, given the small amount of sensors, the fusion delays induced at the base station are negligible and sleep mode is never selected, namely, sensors always transmit.

The learned policy was tested against the two standard design choices *all-raw* and *all-processing* like the previous scenario. The outcome over the horizon is plotted in Fig. 11 and summarized in Table IX. As it is possible to appreciate from Fig. 11, the Q-learning learns to cleverly allocate computational

TABLE IX
MEAN ERROR VARIANCE IN AUTONOMOUS-DRIVING SIMULATION

Q-learning (proposed)	All-raw	All-processing
3.51	3.73	3.69

resources according to the current estimate accuracy. During the transient phase (till 600 ms), when the error variance is large, processing mode is selected for lidar, cameras and one radar, according to the first two rows in Table VIII. Notably, this choice performs close to *all-processing* (red curve), while the *all-raw* configuration is clearly disadvantageous (higher blue curve). Conversely, at steady state only the cameras are in processing mode: this resembles more closely the *all-raw* policy, which performs better (lower blue curve) than *all-processing*.

Overall, we can see from Table IX that the proposed approach leads to a total improvement of about 5% compared to baseline policies. While this result may look marginal, we note that the improvement is rather small over the main transient phase, because the Kalman predictor is able to drop the error variance very quickly for all sensing configurations, but is way larger (about $15-20\%$) when the curves settle about small values. Also, while the objective cost (9a) refers to the whole horizon, we note that in fact the learned policy performs better than the baselines nearly at each point in time, as Fig. 11 shows, with the curve obtained with the Q-learning policy being almost always below the others. Further, the MA is again consistently smaller than both baselines, highlighting an even better performance of the proposed approach with respect to the targeted optimization.

C. Discussion: The Role of Learning in Model-Based Estimation

The exposed simulations suggest that the proposed approach can improve performance of smart sensor networks dealing with estimation tasks as compared to standard design choices with static processing decisions. In particular, the learning-based design exploits observation of the estimation error online to select effective sensing configurations at different points in time, while baselines cannot adapt to transient or steady-state regimes that benefit from different processing allocations.

It is noteworthy that a learning method such as Q-learning can effectively drive the sensing design, leading to improvement with respect to baselines, even with an estimation tool as effective and robust as the Kalman predictor. Indeed, due to optimality of the latter algorithm applied to the chosen dynamical system, one can expect even trivial choices (such as *all-raw* and *all-processing*) to yield acceptable performance. Conversely, it is hard to suggest good heuristics in the present framework, as the performance varies with respect to the system dynamics, delays, and error variances. In particular, an *optimal* design given all available options is far from trivial: even the simplest setup bears a combinatorial problem that quickly makes deriving an optimal solution computationally infeasible. Indeed, submodularity properties that allow to analytically bound suboptimality of greedy algorithms [47] are hard to meet in realistic

scenarios, e.g., under delays, out-of-sequence message arrivals, or multi-rate sensors [61].

Given these premises, the performance improvements obtained via the studied learning method are encouraging not only with regard to the addressed framework, but mostly in supporting the contribution of such tools to general estimation and control tasks, which can benefit from the power of learning to circumvent computational bottlenecks associated with optimization-based design. Hence, rather than looking at the two domains of model-based and data-driven control as mutually exclusive approaches, this work aims to reinforce arguments supporting a unified, best-of-both-worlds framework.

VI. Conclusion

Motivated by smart sensing for Edge Computing, we have proposed an adaptive design that addresses impact of resource-constrained data sampling, processing, and transmission on performance of a monitoring task. Starting from a suitable mathematical model for the considered class of systems, we have tackled the sensing design problem via Q-learning, showing that the learned design can considerably improve performance compared to standard configurations that do not adapt to the time evolution of the system.

Future research avenues are multifold. Besides challenges of Reinforcement Learning (see Section IV-B), model assumptions may be adjusted to address more realistic sensing and communication, as well as different dynamics or control tasks. Also, our approach should be validated with real-world data.

Acknowledgment

References

[1] H. Sun et al., "Review of challenges and research opportunities for voltage control in smart grids," *IEEE Trans. Power Syst.*, vol. 34, no. 4, pp. 2790–2801, Jul. 2019.

[2] H. Erdem and V. Gungor, "On the lifetime analysis of energy harvesting sensor nodes in smart grid environments," *Ad Hoc Netw.*, vol. 75/76, pp. 98–105, 2018.

[3] P. K. Reddy Maddikunta et al., "Unmanned aerial vehicles in smart agriculture: Applications, requirements, and challenges," *IEEE Sensors J.*, vol. 21, no. 16, pp. 17608–17619, Aug. 2021.

[4] K. Haseeb, I. Ud Din, A. Almogren, and N. Islam, "An energy efficient and secure IoT-based WSN framework: An application to smart agriculture," *Sensors*, vol. 20, no. 7, 2020, Art. no. 2081.

[5] A. Bueno, M. Godinho Filho, and A. G. Frank, "Smart production planning and control in the industry 4.0 context: A systematic literature review," *Comput. Ind. Eng.*, vol. 149, 2020, Art. no. 106774.

[6] D. Ivanov, S. Sethi, A. Dolgui, and B. Sokolov, "A survey on control theory applications to operational systems, supply chain management, and industry 4.0," *Annu. Rev. Control*, vol. 46, pp. 134–147, 2018.

[7] M. Krugh and L. Mears, "A complementary cyber-human systems framework for industry 4.0 cyber-physical systems," *Manuf. Lett.*, vol. 15, pp. 89–92, 2018.

[8] B. Taner, R. Bhusal, and K. Subbarao, "A nested robust controller design for interconnected vehicles," in *Proc. AIAA Scitech Forum*, 2020. [Online]. Available: https://arc.aiaa.org/doi/epdf/10.2514/6.2020-0602

[9] V. Giammarino, S. Baldi, P. Frasca, and M. L. D. Monache, "Traffic flow on a ring with a single autonomous vehicle: An interconnected

[10] X. Hou, J. Wang, Z. Fang, Y. Ren, K.-C. Chen, and L. Hanzo, "Edge intelligence for mission-critical 6G services in space-air-ground integrated networks," in *IEEE Netw.*, vol. 36, no. 2, pp. 181–189, Mar./Apr. 2022.

[11] P. A. Rad, D. Hofmann, S. A. Pertuz Mendez, and D. Goehringer, "Optimized deep learning object recognition for drones using embedded GPU," in *Proc. IEEE 26th Int. Conf. Emerg. Technol. Factory Autom.*, 2021, pp. 1–7.

[12] S. Laso, M. Linaje, J. Garcia-Alonso, J. M. Murillo, and J. Berrocal, "Deployment of APIs on Android mobile devices and microcontrollers," in *Proc. IEEE Pervasive Comput. Commun. Workshops*, 2020, pp. 1–3.

[13] S. Li, L. Da Xu, and S. Zhao, "5G Internet of Things: A survey," *J. Ind. Inf. Integr.*, vol. 10, pp. 1–9, 2018.

[14] N. Dal Fabbro, M. Rossi, G. Pillonetto, L. Schenato, and G. Piro, "Model-free radio map estimation in massive MIMO systems via semi-parametric Gaussian regression," *IEEE Wireless Commun. Lett.*, vol. 11, no. 3, pp. 473–477, Mar. 2022.

[15] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. ACM Workshop Mobile Big Data*, 2015, pp. 37–42.

[16] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[17] M. J. O'Grady, D. Langton, and G. M. P. O'Hare, "Edge computing: A tractable model for smart agriculture," *Artif. Intell. Agriculture*, vol. 3, pp. 42–51, Sep. 2019.

[18] S. Chen et al., "Internet of Things based smart grids supported by intelligent edge computing," *IEEE Access*, vol. 7, pp. 74089–74102, 2019.

[19] X. Li, L. Huang, H. Wang, S. Bi, and Y.-J. A. Zhang, "An integrated optimization-learning framework for online combinatorial computation offloading in MEC networks," *IEEE Wireless Commun.*, vol. 29, no. 1, pp. 170–177, Feb. 2022.

[20] H. Xing, O. Simeone, and S. Bi, "Decentralized federated learning via SGD over wireless D2D networks," in *Proc. IEEE 21st Int. Workshop Signal Process. Adv. Wireless Commun.*, 2020, pp. 1–5.

[21] M. M. Amiri, D. Gündüz, S. R. Kulkarni, and H. V. Poor, "Convergence of update aware device scheduling for federated learning at the wireless edge," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3643–3658, Jun. 2021.

[22] M. Chen et al., "Distributed learning in wireless networks: Recent progress and future challenges," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3579–3605, Dec. 2021.

[23] W. Fang, Y. Zhang, B. Yu, and S. Liu, "FPGA-based ORB feature extraction for real-time visual SLAM," in *Proc. IEEE Int. Conf. Field Programmable Technol.*, 2017, pp. 275–278.

[24] Y. Chang, Y. Tian, J. P. How, and L. Carlone, "Kimera-multi: A system for distributed multi-robot metric-semantic simultaneous localization and mapping," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 11210–11218.

[25] M. Anvaripour, M. Saif, and M. Ahmadi, "A novel approach to reliable sensor selection and target tracking in sensor networks," *IEEE Trans. Ind. Informat.*, vol. 16, no. 1, pp. 171–182, Jan. 2020.

[26] L. Mao, L. Jackson, and B. Davies, "Effectiveness of a novel sensor selection algorithm in PEM fuel cell on-line diagnosis," *IEEE Trans. Ind. Electron.*, vol. 65, no. 9, pp. 7301–7310, Sep. 2018.

[27] A. Brunello, A. Urgolo, F. Pittino, A. Montvay, and A. Montanari, "Virtual sensing and sensors selection for efficient temperature monitoring in indoor environments," *Sensors*, vol. 21, no. 8, Jan. 2021, Art. no. 2728.

[28] T. Devos, M. Kirchner, J. Croes, W. Desmet, and F. Naets, "Sensor selection and state estimation for unobservable and non-linear system models," *Sensors*, vol. 21, no. 22, Jan. 2021, Art. no. 7492.

[29] S. Schön et al., "Integrity and collaboration in dynamic sensor networks," *Sensors*, vol. 18, no. 7, Jul. 2018, Art. no. 2400.

[30] F. Li, M. C. De Oliveira, and R. E. Skelton, "Integrating information architecture and control or estimation design," *SICE J. Control Meas. Syst. Integr.*, vol. 1, no. 2, pp. 120–128, Mar. 2008.

[31] E. Clark, S. L. Brunton, and J. N. Kutz, "Multi-fidelity sensor selection: Greedy algorithms to place cheap and expensive sensors with cost constraints," *IEEE Sensors J.*, vol. 21, no. 1, pp. 600–611, Jan. 2021.

[32] M. Alonso, H. Amaris, D. Alcala, and D. M. Florez R., "Smart sensors for smart grid reliability," *Sensors*, vol. 20, no. 8, Jan. 2020, Art. no. 2187.

[33] M. F. Khan, M. Bibi, F. Aadil, and J.-W. Lee, "Adaptive node clustering for underwater sensor networks," *Sensors*, vol. 21, no. 13, Jun. 2021, Art. no. 4514.

[34] S. T. Jawaid and S. L. Smith, "Submodularity and greedy algorithms in sensor scheduling for linear dynamical systems," *Automatica*, vol. 61, pp. 282–288, Nov. 2015.

[35] V. Gupta, T. H. Chung, B. Hassibi, and R. M. Murray, "On a stochastic sensor selection algorithm with applications in sensor scheduling and sensor coverage," *Automatica*, vol. 42, no. 2, pp. 251–260, Feb. 2006.

[36] D. Maity, D. Hartman, and J. S. Baras, "Sensor scheduling for linear systems: A covariance tracking approach," *Automatica*, vol. 136, Feb. 2022, Art. no. 110078.

[37] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. S. Sastry, "Foundations of control and estimation over lossy networks," *Proc. IEEE*, vol. 95, no. 1, pp. 163–187, Jan. 2007.

[38] A. Chiuso, N. Laurenti, L. Schenato, and A. Zanella, "LQG cheap control subject to packet loss and SNR limitations," in *Proc. IEEE Eur. Control Conf.*, 2013, pp. 2374–2379.

[39] S. Tatikonda and S. Mitter, "Control under communication constraints," *IEEE Trans. Autom. Control*, vol. 49, no. 7, pp. 1056–1068, Jul. 2004.

[40] J. L. Ny and G. J. Pappas, "Differentially private filtering," *IEEE Trans. Autom. Control*, vol. 59, no. 2, pp. 341–354, Feb. 2014.

[41] E. Shafieepoorfard and M. Raginsky, "Rational inattention in scalar LQG control," in *Proc. IEEE 52nd Conf. Decis. Control*, 2013, pp. 5733–5739.

[42] G. N. Nair and R. J. Evans, "Stabilizability of stochastic linear systems with finite feedback data rates," *SIAM J. Control Optim.*, vol. 43, no. 2, pp. 413–436, Jan. 2004.

[43] M. Pezzutto, F. Tramarin, S. Dey, and L. Schenato, "Adaptive transmission rate for LQG control over Wi-Fi: A cross-layer approach," *Automatica*, vol. 119, Sep. 2020, Art. no. 109092.

[44] P. Park, S. Coleri Ergen, C. Fischione, C. Lu, and K. H. Johansson, "Wireless network design for control systems: A survey," *IEEE Commun. Surveys Tut.*, vol. 20, no. 2, pp. 978–1013, Second Quarter 2018.

[45] F. Branz, R. Antonello, M. Pezzutto, S. Vitturi, F. Tramarin, and L. Schenato, "Drive-by-Wi-Fi: Model-based control over wireless at 1 kHz," *IEEE Trans. Control Syst. Technol.*, vol. 30, no. 3, pp. 1078–1089, May 2022.

[46] H. Li and Y. Shi, "Network-based predictive control for constrained nonlinear systems with two-channel packet dropouts," *IEEE Trans. Ind. Electron.*, vol. 61, no. 3, pp. 1574–1582, Mar. 2014.

[47] V. Tzoumas, L. Carlone, G. J. Pappas, and A. Jadbabaie, "LQG control and sensing co-design," *IEEE Trans. Autom. Control*, vol. 66, no. 4, pp. 1468–1483, Apr. 2021.

[48] G. Zardini, A. Censi, and E. Frazzoli, "Co-design of autonomous systems: From hardware selection to control synthesis," in *Proc. IEEE Eur. Control Conf.*, 2021, pp. 682–689.

[49] R. Aldana-López, R. Aragüés, and C. Sagüés, "Attention vs. precision: Latency scheduling for uncertainty resilient control systems," in *Proc. IEEE Conf. Decis. Control*, 2020, pp. 5697–5702.

[50] J. Rüth, R. Glebke, T. Ulmen, and K. Wehrle, "Demo abstract: Towards in-network processing for low-latency industrial control," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2018, pp. 1–2.

[51] B. Lin et al., "A time-driven data placement strategy for a scientific workflow combining edge computing and cloud computing," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 4254–4265, Jul. 2019.

[52] T. Taami, S. Krug, and M. O'Nils, "Experimental characterization of latency in distributed IoT systems with cloud fog offloading," in *Proc. IEEE 15th Int. Workshop Factory Commun. Syst.*, 2019, pp. 1–4.

[53] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3056–3069, Nov. 2017.

[54] X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, and R. P. Liu, "Energy-efficient admission of delay-sensitive tasks for mobile edge computing," *IEEE Trans. Commun.*, vol. 66, no. 6, pp. 2603–2616, Jun. 2018.

[55] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1183–1210, May 2021.

[56] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "The cost of delay in status updates and their value: Non-linear ageing," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4905–4918, Aug. 2020.

[57] V. Tripathi, R. Talak, and E. Modiano, "Age optimal information gathering and dissemination on graphs," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 54–68, Jan. 2023.

[58] T. Z. Ornee and Y. Sun, "Sampling and remote estimation for the ornstein-uhlenbeck process through queues: Age of information and beyond," *IEEE/ACM Trans. Netw.*, vol. 29, no. 5, pp. 1962–1975, Oct. 2021.

[59] V. Tripathi, L. Ballotta, L. Carlone, and E. Modiano, "Computation and communication co-design for real-time monitoring and control in multi-agent systems," in *Proc. IEEE 19th Int. Symp. Model. Optim. Mobile Ad Hoc Wireless Netw.*, 2021, pp. 1–8.

[60] I. Kadota and E. Modiano, "Minimizing the age of information in wireless networks with stochastic arrivals," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 1173–1185, Mar. 2021.

[61] L. Ballotta, L. Schenato, and L. Carlone, "Computation-communication trade-offs and sensor selection in real-time estimation for processing networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2952–2965, Fourth Quarter 2020.

[62] L. Ballotta, G. Peserico, and F. Zanini, "A reinforcement learning approach to sensing design in resource-constrained wireless networked control systems," in *Proc. IEEE 61st Conf. Decis. Control*, 2022, pp. 3914–3919.

[63] Z. Shi et al., "Artificial intelligence techniques for stability analysis and control in smart grids: Methodologies, applications, challenges and future directions," *Appl. Energy*, vol. 278, Nov. 2020, Art. no. 115733.

[64] G. Baggio, D. S. Bassett, and F. Pasqualetti, "Data-driven control of complex networks," *Nature Commun.*, vol. 12, no. 1, Mar. 2021, Art. no. 1429.

[65] S. Wang et al., "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 63–71.

[66] H. Medeiros, J. Park, and A. Kak, "Distributed object tracking using a cluster-based Kalman filter in wireless camera networks," *IEEE J. Sel. Top. Signal Process.*, vol. 2, no. 4, pp. 448–463, Aug. 2008.

[67] H. Yang, K. Zhang, K. Zheng, and Y. Qian, "Leveraging linear quadratic regulator cost and energy consumption for ultrareliable and low-latency IoT control systems," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8356–8371, Sep. 2020.

[68] H. Jeon and Y. Eun, "A stealthy sensor attack for uncertain cyber-physical systems," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6345–6352, Aug. 2019.

[69] V. Radisavljevic-Gajic, S. Park, and D. Chasaki, "Vulnerabilities of control systems in Internet of Things applications," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1023–1032, Apr. 2018.

[70] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: Bridging the gap via the real-time iteration," *Int. J. Control*, vol. 93, no. 1, pp. 62–80, Jan. 2020.

[71] M.-C. Tsai and D.-W. Gu, *Robust and Optimal Control*. London, U.K.: Springer, 2014.

[72] P. Hovareshti, V. Gupta, and J. S. Baras, "Sensor scheduling using smart sensors," in *Proc. IEEE 46th Conf. Decis. Control*, 2007, pp. 494–499.

[73] V. Gupta, N. C. Martins, and J. S. Baras, "Optimal output feedback control using two remote sensors over erasure channels," *IEEE Trans. Autom. Control*, vol. 54, no. 7, pp. 1463–1476, Jul. 2009.

[74] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 405–421.

[75] R. Tou and J. Zhang, "IMM approach to state estimation for systems with delayed measurements," *IET Signal Process.*, vol. 10, no. 7, pp. 752–757, 2016.

[76] A. Gopalakrishnan, N. S. Kaisare, and S. Narasimhan, "Incorporating delayed and infrequent measurements in extended Kalman filter based nonlinear state estimation," *J. Process Control*, vol. 21, no. 1, pp. 119–129, 2011.

[77] L. Greco, D. Fontanelli, and A. Bicchi, "Design and stability analysis for anytime control via stochastic scheduling," *IEEE Trans. Autom. Control*, vol. 56, no. 3, pp. 571–585, Mar. 2011.

[78] M. Pavlidakis, S. Mavridis, N. Chrysos, and A. Bilas, "TReM: A task revocation mechanism for GPUs," in *Proc. IEEE 22nd Int. Conf. High Perform. Comput. Commun.; IEEE 18th Int. Conf. Smart City; IEEE 6th Int. Conf. Data Sci. Syst.*, 2020, pp. 273–282.

[79] H. Lee and J. Lee, "Limited non-preemptive EDF scheduling for a real-time system with symmetry multiprocessors," *Symmetry*, vol. 12, no. 1, 2020, Art. no. 172.

[80] T. Fedullo, A. Morato, F. Tramarin, L. Rovati, and S. Vitturi, "A comprehensive review on time sensitive networks with a special focus on its applicability to industrial smart and distributed measurement systems," *Sensors*, vol. 22, no. 4, 2022, Art. no. 1638.

[81] W. Yu, M. Jia, C. Liu, and Z. Ma, "Task preemption based on petri nets," *IEEE Access*, vol. 8, pp. 11512–11519, 2020.

[82] T. Jaakkola, M. Jordan, and S. Singh, "Convergence of stochastic iterative dynamic programming algorithms," in *Proc. 6th Int. Conf. Neural Inf. Process. Syst.*, 1993, pp. 703–710.

[83] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan, "Is Q-learning provably efficient?," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4868–4878.

[84] G. Li, C. Cai, Y. Chen, Y. Gu, Y. Wei, and Y. Chi, "Tightening the dependence on horizon in the sample complexity of Q-learning," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 6296–6306.

[85] S. Chinchali et al., "Network offloading policies for cloud robotics: A learning-based approach," *Auton. Robots*, vol. 45, no. 7, pp. 997–1012, 2021.

[86] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C., Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[87] A. Allan, "The big benchmarking roundup," Aug. 2019. [Online]. Available: https://aallan.medium.com/the-big-benchmarking-roundup-a561fbfe8719

[88] S. Hossain and D.-J. Lee, "Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with GPU-Based embedded devices," *Sensors*, vol. 19, no. 15, Jan. 2019, Art. no. 3371.

[89] T. Dalsa, "Genie nano series datasheet," 2021. [Online]. Available: https://www.stemmer-imaging.com/media/uploads/cameras/dalsa/12/122239-Teledyne-DALSA-Genie-Nano-Series-Manual.pdf

[90] M. Casares, A. Pinto, Y. Wang, and S. Velipasalar, "Power consumption and performance analysis of object tracking and event detection with wireless embedded smart cameras," in *Proc. IEEE 3rd Int. Conf. Signal Process. Commun. Syst.*, 2009, pp. 1–8.

[91] A. Barreiro, A. Baños, and E. Delgado, "Reset control of the double integrator with finite settling time and finite jerk," *Automatica*, vol. 127, May 2021, Art. no. 109536.

[92] H. G. Oral, E. Mallada, and D. Gayme, "On the role of interconnection directionality in the quadratic performance of double-integrator networks," *IEEE Trans. Autom. Control*, vol. 67, no. 11, pp. 6211–6218, Nov. 2022.

[93] V. G. Rao and D. S. Bernstein, "Naive control of the double integrator," *IEEE Control Syst. Mag.*, vol. 21, no. 5, pp. 86–97, Oct. 2001.

[94] W. Ren, "On consensus algorithms for double-integrator dynamics," *IEEE Trans. Autom. Control*, vol. 53, no. 6, pp. 1503–1509, Jul. 2008.

[95] D. Feng et al., "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1341–1360, Mar. 2021.

[96] Y. Li et al., "Deep learning for LiDAR point clouds in autonomous driving: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3412–3432, Aug. 2021.

[97] S. Sun and Y. D. Zhang, "4D automotive radar sensing for autonomous vehicles: A sparsity-oriented approach," *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 4, pp. 879–891, Jun. 2021.

[98] B.-J. Kim and S.-B. Lee, "Safety evaluation of autonomous vehicles for a comparative study of camera image distance information and dynamic characteristics measuring equipment," *IEEE Access*, vol. 10, pp. 18486–18506, 2022.

[99] S. Sun, A. P. Petropulu, and H. V. Poor, "MIMO radar for advanced driver-assistance systems and autonomous driving: Advantages and challenges," *IEEE Signal Process. Mag.*, vol. 37, no. 4, pp. 98–117, Jul. 2020.

[100] L. van Dijk, "Future vehicle networks and ECUs architecture and technology considerations," 2017. [Online]. Available: https://www.nxp.com/docs/en/white-paper/FVNECUA4WP.pdf

[101] Omnet++. 2020. [Online]. Available: https://omnetpp.org/

Giovanni Peserico received the master's degree in automation engineering and the Ph.D. degree in "Alto Apprendistato" in information Engineering in collaboration with Autec S.R.L from the University of Padova, Padova, Italy, in 2019 and 2023, respectively. He is currently covering a Cybersecurity Software Engineer position for Qascom, an Italian company specialized in GNSS authentication and space cybersecurity. His research interests include safety and cybersecurity, industrial and wireless networks, networked control systems, and learning-based safe control.

Francesco Zanini received the master's degree in automation engineering and the Ph.D. degree in information engineering from the University of Padova, Padova, Italy, in 2019 and 2023, respectively. He was a Visiting Student with the University of Alberta, Edmonton, AB, Canada, in 2022, and later joined the institution as a Postdoctoral Researcher in 2023. His research interests include the intersection of reinforcement learning and dynamical systems, along with Koopman operators and learning theory.

Paolo Dini received the M.Sc. and Ph.D. degrees from the Università di Roma La Sapienza, Rome, Italy, in 2001 and 2005, respectively. He is currently a Senior Researcher with the Centre Tecnológic de Telecomunicacions de Catalunya (CTTC), Castelldefels, Spain, where he coordinates the activities of the Sustainable Artificial Intelligence research unit. His research interests include sustainable computing and networking, distributed optimization and machine learning, multi-agent systems and decision-making processes, data mining for cyber-physical systems.

He is involved in more than 25 research projects during his career. He is currently the Coordinator of the CHIST-ERA SONATA Project on sustainable computing and communication at the edge and the Scientific Coordinator of the MSCA Greenedge European Training Network on edge intelligence and sustainable computing. His research activity is documented in more than 90 peer-reviewed scientific journals and international conference papers. He was the recipient of the two awards from the Cisco Silicon Valley Foundation for his research on heterogeneous mobile networks in 2008 and 2011, respectively. He has co-organized several training events and workshops/special sessions at several international conferences sponsored by IEEE. He is a TPC in many international conferences and a Reviewer of several scientific journals of the IEEE, Elsevier, ACM, Springer. He has been European Climate Pact Ambassador since 2022 and participates in several outreach events (e.g., Research Nights) to promote sustainable design principles.

Luca Ballotta (Member, IEEE) received the master's degree in automation engineering and the Ph.D. degree in information engineering from the University of Padova, Padova, Italy, in 2019 and 2023, respectively. He was a Visiting Student with the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2020 and 2022. His research interests include multi-agent systems and networked control systems subject to resource constraints, resilient distributed optimization, and learning-based safe control. He was the recipient of the Young Author Prize at the 2020 IFAC World Congress.