EFFICIENT DISTRIBUTION OF VISUAL PROCESSING TASKS IN MULTI-CAMERA VISUAL SENSOR NETWORKS

Emil Eriksson, Valentino Pacifici, György Dán

School of Electrical Engineering, KTH Royal Institute of Technology, Stockholm, Sweden
{emieri,pacifici,gyuri}@kth.se

ABSTRACT

Multi-camera visual sensor networks (VSNs) require large computational resources in order to perform visual analysis in real-time. One way to match the computational needs is to augment the VSN with dedicated processing nodes that do in-network processing, but this requires careful allocation of loads from the sensor nodes in order to ensure low processing times. In this paper we formulate the problem of load allocation and completion time minimization in a VSN as an optimization problem. We propose a distributed algorithm for load allocation, and evaluate its performance in terms of completion time and convergence compared to a *Greedy* algorithm. Simulations show that the proposed algorithm converges faster, but at the cost of increased completion times. Nonetheless, combined with appropriate coordination, the proposed algorithm achieves low completion times at low complexity.

Index Terms— Visual feature extraction, Sensor networks, Divisible load theory, Distributed optimization

1. INTRODUCTION

Real-time visual analysis in visual sensor networks (VSNs) typically requires large computational capacity in order to perform the visual processing tasks within the specified time limit. As shown in [1], significant delay is incurred both when processing is performed at the sensor nodes, and when images are transmitted across the network to a central processing node. Transmitting the images across the network may also drain the energy resources of nodes relaying the images. As low cost sensor nodes are becoming available, introducing dedicated processing nodes for distributed in-network processing has been proposed to increase the computational capacity available to the sensor nodes [2, 3].

Visual analysis involves several processing steps. First, a detection filter is applied at every pixel of an image in order to determine whether the pixel is an interest point or not. Second, for each detected interest point, a feature descriptor is extracted based on the region surrounding the interest point (examples include [4, 5, 6, 7, 8]). The set of feature descriptors can then be used to perform the application task, e.g. object recognition, or tracking. By dividing the image into multiple sub-areas, the interest point detection and feature extraction tasks can be divided among the processing nodes, this method is referred to as area-split in [9]. To achieve the best performance, the size and allocation of the sub-areas must be chosen so as to minimize the completion times of the processing nodes, which is achieved when all processors finish at the same time [10]. However, as the number of interest points detected in an image, as well as their spatial distribution, depends both on the parameters of the applied detection method, and on the visual contents of the image, which is not known a priori, the completion times of the processing nodes can not be determined solely based on the size of the sub-area assigned to the node. Additionally, if multiple sensors are sharing a set of processing nodes, they will compete for the use of those processing nodes. Considering the randomness of the visual content, as well as the interaction of the sensor nodes, creating efficient distributed algorithms for completion time minimization of distributed processing tasks in VSNs is a challenging task.

In this paper we address the completion time minimization problem in a VSN equipped with multiple visual sensor nodes and processing nodes. We propose a coordinated and a fully distributed algorithm for assigning sub-areas to processing nodes,and we prove convergence of the proposed algorithm for the scenario when only one sensor updates its allocation at a time. We use simulations to evaluate the performance and complexity of the algorithm in comparison to a *Greedy* algorithm based on a surveillance video trace. Our results show that the proposed algorithm converges to a stable allocation faster than the *Greedy* algorithm, but results in slightly higher completion times. For the scenario where sensors are not allowed to converge upon each frame, but can only make a single update, the completion time can be decreased by coordinating the assignments of the sensor nodes .

The rest of the paper is organized as follows. In Section 2 we describe the considered system and in Section 3 we formulate the problem of completion time minimization. In Section 4 we present and analyze algorithms for solving the completion time minimization problem. In Section 5 we present numerical results and we conclude the paper in Section 6.

2. SYSTEM MODEL

Our system model is similar to that in [11]. We consider a visual sensor network (VSN) consisting of a set of sensor nodes $\mathcal{S}$, $|\mathcal{S}| = S$, and a set of processing nodes $\mathcal{N}$, $|\mathcal{N}| = N$. Sensor node $s \in \mathcal{S}$ captures a sequence $\mathcal{I}_s = \{1, ...\}$ of images of width w pixels. For the delegation of the computation, sensor node s divides image i into $V_s^i \leq N$ vertical slices. This

The project GreenEyes acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 296676.

scheme was referred to as area-split in [12, 9]. We define slice v using its normalized leftmost and rightmost horizontal coordinates, $x^i_{s,v-1}$ and $x^i_{s,v}$, i.e., $x^i_{s,0} = 0$ and $x^i_{s,V^i_s} = 1$, and we define the cutpoint location vector for image i as $x^i_s = \{x^i_{s,0}, \ldots, x^i_{s,V^i_s}\}$. For convenience, we use $y^i_{s,v} = x^i_{s,v} - x^i_{s,v-1}$ to denote the normalized width of slice v, and we define $y^i_{s,v} = 0$ for $v \leq 0$ and for $v > V^i_s$. Thus, by definition, $\sum_{v=1}^{V^i_s} y^i_{s,v} = 1$. Sensor s transmits slice $1 \leq v \leq V^i_s$ to processing node $d^i_s(v) \in \mathcal{N}$ for processing. We define d^i_s as a sequence of $|d^i_s| = V^i_s$ distinct elements, and with slight abuse of notation we use $n \in d^i_s$ if $d^i_s(v) = n$ for some $1 \leq v \leq V^i_s$, i.e., node n is used by sensor s. We use the notation $\mathcal{p}^i_s(n)$ for the slice that sensor s assigns to node n. We refer to d^i_s as the assignment by sensor s, and to $\mathcal{p}^i_s$ as its inverse.

2.1. Visual feature extraction

Each processing node n computes local visual features from the image slices assigned to it. The computation of local features starts with interest point detection in an image, by applying a blob detector or an edge detector at every pixel of an image [6, 7, 8]. For each pixel, the detector computes a response score based on a square area centered around it. We denote the side length of the square normalized by the width of the image by $2o$. The side length $2ow$ of the square (in pixels) depends on the applied detector. Therefore, for correct operation, each slice v transmitted to processing node n has to be appended on one or both sides by an overlap area of width o.

The time it takes to detect interest points can be modeled as a linear function of image size in pixels and of the number $\xi^i_{s,v} = f_i(x^i_{s,v}, x^i_{s,v-1})$ of interest points detected; this model was validated recently in [1, 2]. We can thus model the detection time for slice v from sensor s at processing node n as a function of the image slice width $y^i_{s,v}$ and of the number $\xi^i_{s,v}$ of interest points detected in the image slice as an affine function $P_n(y^i_{s,v} + \alpha_f \xi^i_{s,v})$, where P_n is the per unit processing time of node n. Note that $\xi^i_{s,v}$ is unknown before processing image slice v, but efficient low-complexity predictors exist, such as the last value predictor [2].

After detection, a feature descriptor is extracted for each interest point by comparing pixel intensities. The time it takes to extract the descriptors can be modeled as a linear function of the number $\xi^i_{s,v}$ of interest points detected, as shown in [2]. We can thus model the detection and extraction time as $P_n(y^i_{s,v} + (\alpha_f + \alpha_e)\xi^i_{s,v}) = P_n(y^i_{s,v} + \alpha_d \xi^i_{s,v})$. We consider that α_f, α_e and thus α_d are the same for all processing nodes, which is reasonable if the nodes have a similar computer architecture (e.g., instruction set).

In applications where features extracted from images captured by multiple cameras are needed, e.g., in the case of multi-camera tracking for updating a hidden-Markov model or a particle filter, the extraction of the features from all images should finish at the same time. We thus consider that if a processing node n has to process image slices from different sensors simultaneously, then it allocates its processing power in a way that ensures that the processing of all slices is completed at the same time.

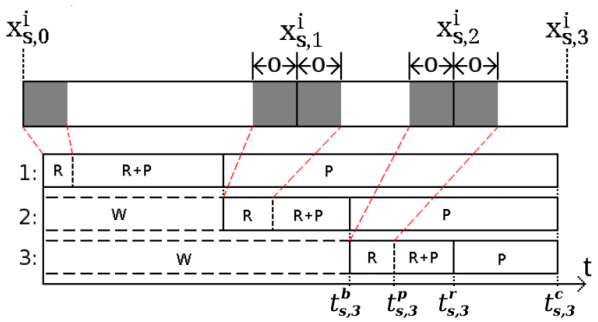

Fig. 1: Transmission and processing schedule of slices from a single sensor to three processing nodes. 'R' stands for receiving the image slice, 'P' for processing, and 'W' for waiting.

2.2. Communication Model

The nodes communicate using a wireless communication protocol, such as IEEE 802.15.4 or IEEE 802.11, in which transmissions suffer from packet losses due to wireless channel impairments. As measurement studies show [13, 14], the loss burst lengths at the receiver have low mean and variance in the order of a couple of frames [15], [16]. Therefore, a widely used model of the loss process is a low-order Markov-chain, with fast decaying correlation and short mixing time. In the system we consider, the amount of data to be transmitted to the processing nodes is relatively large, and therefore it is reasonable to model the average transmission time from sensor s to processing node n as a linear function of the amount of transmitted data. We denote the transmission time coefficient by $C_{s,n}$, which can be interpreted as the average per image transmission time, including potential retransmissions. As the throughput is close to stationary over short timescales, $C_{s,n}$ can be estimated [17]. When there are several sensors transmitting data, the MAC protocol provides airtime fairness for the transmitters [18], thus the actual transmission time coefficient is proportional to the number of sensors transmitting. For example, when there are S sensors transmitting, the actual transmission time coefficient is $SC_{s,n}$. Motivated by the rapid increase of 802.11 wireless transmission capacities, we consider that processing is slower than transmission, i.e., $min_n P_n(1 + \alpha_d) \geq S \max_{s,n} C_{s,n}$, and thus processing an image slice takes at least as much time as receiving it.

The processing nodes can receive data and perform processing simultaneously. Thus, a processing node can start processing slice v from sensor s after it has received o (for $v = 1$) and $2o$ (for $1 < v \leq V^i_s$) worth of data. The resulting regions of overlap in adjacent slices could in principle be transmitted in multicast to the appropriate processing nodes, but experimental results show that multicast transmission suffers from low throughput in practice due to lack of link layer retransmissions and missing channel quality information [19], we thus consider that all data transmissions are done using unicast.

3. COMPLETION TIME AND PROBLEM FORMULATION

Using the model of transmission and processing above, let us consider the completion time of the processing of image $i \in \mathcal{I}_s$ captured by sensor s. Figure 1 illustrates the transmission and processing of slices to $N = 3$ processing nodes. Let us denote

by $t^b_{s,v}(\mathbf{x^i},\mathbf{d^i})$ the time instant when processing node $d^i_s(v)$ receives the first bit of slice v from sensor s, by $t^p_{s,v}(\mathbf{x^i},\mathbf{d^i})$ the time instant when processing of slice v from sensor s starts at processing node $d^i_s(v)$ (i.e., after receiving the overlap), by $t^r_{s,v}(\mathbf{x^i},\mathbf{d^i})$ the time instant when processing node $d^i_s(v)$ receives the last bit of slice v from sensor s, and by $t^c_{s,v}(\mathbf{x^i},\mathbf{d^i})$ the time instant when processing of slice v from sensor s is completed at processing node $d^i_s(v)$. If not otherwise specified, we use the shorthand $t^r_{s,v}$ to refer to $t^r_{s,v}(\mathbf{x^i},\mathbf{d^i})$.

Observe that the time $t^r_{s,v} - t^b_{s,v}$ it takes node s to transmit slice v to processing node n depends on the number of sensor nodes that are transmitting simultaneously. To capture the dependence of the transmission time on the sensors' cutpoint location vectors $(x^i_s)_{s\in\mathcal{S}}$ and assignment functions $(d^i_s)_{s\in\mathcal{S}}$, we define the experienced transmission time coefficient

$$\tilde{C}_{s,n}(\mathbf{x^i},\mathbf{d^i}) = \begin{cases} (t^r_{s,v} - t^b_{s,v})/(y^i_{s,v}+o) & \text{for } v = 1, V^i_s \\ (t^r_{s,v} - t^b_{s,v})/(y^i_{s,v}+2o) & \text{for } 1 < v < V^i_s. \end{cases}$$

Similarly, the time it takes processing node n to complete the processing of slice v sent by sensor s depends on whether or not the processing node has to process slices from other sensors simultaneously. We define the experienced processing time coefficient of sensor s at processing node n as $\tilde{P}_{s,n}(\mathbf{x^i},\mathbf{d^i}) = (t^c_{s,v} - t^p_{s,v})/(y^i_{s,v} + \alpha_d \xi^i_{s,v})$.

We can express the mean completion time of slice v delegated by sensor s to processing node $n = d^i_s(v)$ as a function of the experienced transmission time coefficients and of the experienced processing time coefficients. For the first slice, i.e., $n = d^i_s(1)$, we have

$$T^i_{s,n}(\mathbf{x^i},\mathbf{d^i}) = \tilde{C}_{s,n}(\mathbf{x^i},\mathbf{d^i})o + \tilde{P}_{s,n}(\mathbf{x^i},\mathbf{d^i})(y^i_{s,1} + \alpha_d \xi^i_{s,1}). \quad (1)$$

For the remaining slices, i.e., $n = d^i_s(v), v > 1$, the completion time depends also on the transmission times of previous slices

$$\begin{aligned} T^i_{s,n}(\mathbf{x^i},\mathbf{d^i}) &= \tilde{C}_{s,d^i_s(1)}(\mathbf{x^i},\mathbf{d^i})[y^i_{s,1}+o] + \tilde{C}_{s,n}(\mathbf{x^i},\mathbf{d^i})2o \\ &+ \textstyle\sum_{\nu=2}^{v-1} \tilde{C}_{s,d^i_s(\nu)}(\mathbf{x^i},\mathbf{d^i})[y^i_{s,\nu}+2o] \\ &+ \tilde{P}_{s,n}(\mathbf{x^i},\mathbf{d^i})(y^i_{s,v} + \alpha_d \xi^i_{s,v}). \end{aligned} \quad (2)$$

Finally, we define the completion time of image i for sensor s as the completion time of the processing node that finishes last

$$T^i_s(\mathbf{x^i},\mathbf{d^i}) = \max_{n\in d^i_s}(T^i_{s,n}(\mathbf{x^i},\mathbf{d^i})). \quad (3)$$

Observe that the maximum is taken only over the processing nodes used by sensor s.

3.1. Completion Time Minimization

Given the set of sensor nodes $\mathcal{S}$, the set of processing nodes $\mathcal{N}$, the transmission and processing time coefficients $C_{s,n}$ and P_n, we can formulate the problem of minimizing the completion time for image i as a combinatorial optimization problem

$$\min_{(\mathbf{x^i},\mathbf{d^i})} t \qquad \text{s.t.} \quad (4)$$

$$T^i_s(\mathbf{x^i},\mathbf{d^i}) \leq t \ \ \forall s \in \mathcal{S} \quad (5)$$

$$x^i_{s,v-1} - x^i_{s,v} \leq -o \ \ 1 \leq v \leq V^i_s \quad (6)$$

$$x^i_{s,v} w \in \{1,\ldots,w\} \ \ 1 \leq v \leq V^i_s \quad (7)$$

where w is the width of the image in pixels. Observe that the sensors may not have sufficient information to solve the optimization problem, e.g., because the interest point distribution is unknown before processing an image. Nonetheless, as the interest point distribution changes slow enough an approximate solution can be obtained using the interest point distribution of the previous image.

Solving the optimization problem is, however, computationally demanding, as it requires one to consider all $\sum_{n=1}^{N} \frac{N!}{(N-n)!}$ profiles of partial permutations $(d^i_s)_{s\in\mathcal{S}}$, and for each profile find the optimal allocation vector $\mathbf{x^i}$. Thus, given the computational constraints in the sensors it may be infeasible to solve even moderate instances of the problem.

We thus consider that there is an entity, e.g., in a mobile edge-cloud, that has sufficient computational resource for computing the optimal assignment $\bar{d}$ and and can communicate it to the sensor nodes periodically. Whether or not this entity is used by the VSN determines the operation mode of the VSN.

Definition 1. VSN Operation Mode: *In the* un-coordinated assignment *mode each sensor s is allowed to update (d^i_s, x^i_s) upon each frame i. In the* coordinated assignment *mode, the assignment $\bar{d}$ is periodically computed by a central entity, e.g., a cloud-based resource, after every R frames and is communicated to the sensors; the sensors can update (x^i_s) only. We refer to R as the inter-refresh time under coordinated assignment.*

4. DISTRIBUTED ALGORITHMS

In the following we describe a fully distributed algorithm for computing the sub-area allocations of the sensors to processing nodes, based on information they can obtain via measurements and through signaling.

We refer to the times when the sensors can revise their allocations as the *revision opportunity*, which can be either synchronous or asynchronous.

Definition 2. Revision opportunity: *Under asynchronous revision only one sensor $s \in \mathcal{S}$ is allowed to update its allocation upon each image i. Under synchronous revision every sensor is allowed to update its allocation upon every image i.*

While in a VSN synchronous revision is easy to implement, asynchronous revision could, e.g., be implemented by configuring a static revision order through modulo division of the image sequence number.

In Fig. 2 we describe a distributed algorithm that allows each sensor s to update its allocation if doing so does not increase the sensors' transmission times, called the Transmission Time Preserving (*TTP*) algorithm. The set of sensors that are allowed to update their allocation upon image i are denoted by $D_i \subseteq \mathcal{S}$. Observe that $|D_i| = 1$ for asynchronous and $D_i = \mathcal{S}$ for synchronous revision opportunities, respectively.

Observe that $(t^b_{s',\mathrm{p}^i_{s'}(n)} - t^r_{s',\mathrm{p}^i_{s'}(n)})$ is a known linear function of $y^i_{s',\mathrm{p}^i_{s'}(n)}$ and of the transmission time coefficient $C_{s',n}$, and thus every sensor s can compute $C_{s',n}$ for $n \in d^i_{s'}$. The sensors can also compute the time $t^p_{s',\mathrm{p}^i_{s'}(n)}$ for every processing node n.

Upon image i:

1. Each sensor $s \in D_i$ computes allocation (x'_s, d'_s) s.t.
$$(x'_s, d'_s) = \arg\min_{(x_s, d_s)} T_s((x_s, \mathbf{x}^{\mathbf{i}}_{-\mathbf{s}}), (d_s, \mathbf{d}^{\mathbf{i}}_{-\mathbf{s}})) \quad (8)$$
2. Each sensor $s \in D_i$ revises its assignment from (x^i_s, d^i_s) to $(x^{i+1}_s, d^{i+1}_s) = (x'_s, d'_s)$ only if
$$t^r_{s,|d'_s|}((x'_s, \mathbf{x}^{\mathbf{i}}_{-\mathbf{s}}), (d'_s, \mathbf{d}^{\mathbf{i}}_{-\mathbf{s}})) \leq t^r_{s,V^i_s}(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}}) \quad (9)$$
3. Upon completion, every processing node n broadcasts to each sensor s its processing time coefficient P_n, and the times $(t^b_{s',\mathrm{p}^i_{s'}(n)}, t^r_{s',\mathrm{p}^i_{s'}(n)})$ and the corresponding slice widths $y^i_{s',\mathrm{p}^i_{s'}(n)}$ for all sensors that used node n, i.e., $s' \in \{s' | \exists v \text{ s.t. } d^i_{s'}(v) = n\}$.

Fig. 2: Pseudo-code of Transmission Time Preserving (*TTP*)

4.1. Convergence analysis

If the image contents would not change, a distributed algorithm would provide constant system performance if it reaches an allocation at which the sensors would settle. Such an allocation, if it exists, could be reached either through repeated information exchange between the sensors, or the sensors could compute the allocation if they have sufficient information available (i.e., they can compute the revisions of the other sensors). The first approach involves communication overhead, while the second approach requires information to be available and it increases the computational load of the sensors.

In the following we show that TTP with *asynchronous revision* would settle in a stable allocation. For the analysis we assume that the interest points are evenly spaced along the horizontal axis in every image, we can thus let $\alpha_d = 0$. Furthermore, we assume that each sensor can measure its transmission and processing time coefficients. For notational convenience we omit the index i whenever the expected transmission time and processing time coefficients are used. We start by proving the following lemma.

Lemma 1. *Consider the allocation (x'_s, d'_s) that minimizes the completion time of sensor s given the other sensors' allocation upon image i, given in (8). If there exists a sensor $q \in \mathcal{S} \setminus \{s\}$ such that $T_q((x_s, \mathbf{x}^{\mathbf{i}}_{-\mathbf{s}}), (d_s, \mathbf{d}^{\mathbf{i}}_{-\mathbf{s}})) > T_q(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}})$ and $T_q(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}}) > T_s(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}})$, then*
$$t^r_{s,|d_s|}(x_s, \mathbf{x}^{\mathbf{i}}_{-\mathbf{s}}), (d_s, \mathbf{d}^{\mathbf{i}}_{-\mathbf{s}})) > t^r_{s,V^i_s}(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}}) \quad (10)$$

Proof. We prove the lemma by contradiction. Assume that $t^r_{s,|d_s|}(x_s, \mathbf{x}^{\mathbf{i}}_{-\mathbf{s}}), (d_s, \mathbf{d}^{\mathbf{i}}_{-\mathbf{s}})) \leq t^r_{s,V^i_s}(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}})$. It follows that $\tilde{C}_{q,v}((x_s, \mathbf{x}^{\mathbf{i}}_{-\mathbf{s}}), (d_s, \mathbf{d}^{\mathbf{i}}_{-\mathbf{s}})) \leq \tilde{C}_{q,v}(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}})$ for all $v \in V_q$. Call $\bar{n}$ the processing node where sensor q experiences the slowest completion time, i.e. $\bar{n} = \arg\max_{n \in d_q} T^n_q(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}})$. By assumption $T_{q,\bar{n}}((x_s, \mathbf{x}^{\mathbf{i}}_{-\mathbf{s}}), (d_s, \mathbf{d}^{\mathbf{i}}_{-\mathbf{s}})) > T_{q,\bar{n}}(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}})$. Hence, from the definition of completion time in (2), it follows that that the processing coefficient $P_{q,\bar{n}}((x_s, \mathbf{x}^{\mathbf{i}}_{-\mathbf{s}}), (d_s, \mathbf{d}^{\mathbf{i}}_{-\mathbf{s}})) > P_{q,\bar{n}}(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}})$. Therefore the processing of sensors s and q under allocation $(x_s, \mathbf{x}^{\mathbf{i}}_{-\mathbf{s}}), (d_s, \mathbf{d}^{\mathbf{i}}_{-\mathbf{s}})$ overlaps at node $\bar{n}$, which implies $T_{q,\bar{n}}((x_s, \mathbf{x}^{\mathbf{i}}_{-\mathbf{s}}), (d_s, \mathbf{d}^{\mathbf{i}}_{-\mathbf{s}})) = T_{s,\bar{n}}((x_s, \mathbf{x}^{\mathbf{i}}_{-\mathbf{s}}), (d_s, \mathbf{d}^{\mathbf{i}}_{-\mathbf{s}}))$. From the assumption $T_q(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}}) > T_s(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}})$ it follows that
$$T_s(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}}) < T_{q,\bar{n}}(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}}) < T_{q,\bar{n}}((x_s, \mathbf{x}^{\mathbf{i}}_{-\mathbf{s}}), (d_s, \mathbf{d}^{\mathbf{i}}_{-\mathbf{s}})),$$
which implies $T_s(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}}) < T_{s,\bar{n}}((x_s, \mathbf{x}^{\mathbf{i}}_{-\mathbf{s}}), (d_s, \mathbf{d}^{\mathbf{i}}_{-\mathbf{s}}))$ and contradicts (8). This proves the lemma. □

We are now ready to prove the following.

Theorem 1. TTP *with* asynchronous revision *terminates in a allocation $(\bar{\mathbf{x}}, \bar{\mathbf{d}})$ after a finite number of revisions.*

Proof. For an allocation $(\mathbf{x}, \mathbf{d})$ let us define the $|\mathcal{S}|$-length vector $\tau(\mathbf{x}, \mathbf{d}) = (T_{s_1}, T_{s_2}, \ldots, T_{s_{|\mathcal{S}|}})$ of completion times sorted in decreasing order, i.e., $T_{s_k}(\mathbf{x}, \mathbf{d}) \geq T_{s_{k+1}}(\mathbf{x}, \mathbf{d})$. Consider the revision of sensor s upon image i from assignment (x^i_s, d^i_s) to $(x^{i+1}_s, d^{i+1}_s) = (x'_s, d'_s)$. It follows from (9) and from Lemma 1 that no sensor $q \in \mathcal{S} \setminus \{s\}$ such that $T_q(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}}) > T_s(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}})$ will experience an increase in completion time, i.e. $T_q((x_s, \mathbf{x}^{\mathbf{i}}_{-\mathbf{s}}), (d_s, \mathbf{d}^{\mathbf{i}}_{-\mathbf{s}})) \leq T_q(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}})$. Thus, for the revision of sensor s upon image i it holds $\tau(\mathbf{x}^{\mathbf{i+1}}, \mathbf{d}^{\mathbf{i+1}}) <_L \tau(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}})$, where $<_L$ stands for *lexicographically smaller*. Among all vectors τ of ordered completion times there is a vector that is lexicographically minimal, the vector that corresponds to all sensors completing at the same time. Thus, the algorithm terminates in an allocation $(\bar{\mathbf{x}}, \bar{\mathbf{d}})$ compared to which no sensor can revise its assignment. □

Observe that the result in Theorem 1 cannot be extended to the case of *synchronous revision*. Furthermore, observe that condition (9) in *TTP* prevents the revision by sensor s upon image i to cause an increase of the completion time of any sensor q whose completion time is longer than that of sensor s. While it is clear that condition (9) facilitates the convergence of *TTP*, it is unclear how it affects the system performance. We address this question in the following section.

5. NUMERICAL RESULTS

We evaluate the convergence properties and the completion time of the proposed algorithm through simulations using two video traces. The video traces have a resolution of 720×480 pixels, and frame rates of 30 frames per second. Each simulation is run on the first 300 frames of the video traces. Both traces depict the same group of 9 people, slowly moving around an empty parking lot, with the point of view rotated by approximately $90°$. The observed scene changes very slowly compared to the frame rate, and thus the variation in the interest point distribution is low across the images in the traces.

For the simulations we consider a topology consisting of $S = 2$ sensor nodes and $N = 3$ processing nodes. All the nodes are placed on a two-dimensional plane. The processing nodes are placed between the two sensor nodes such that each processing node is closer to sensor node 1 than sensor node 2. Using the Friis transmission equation to estimate the power at the receiving nodes, the Shannon capacities of the channels can be calculated and gives the transmission time coefficients
$$C = 10^{-2} \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \end{pmatrix} s/image$$

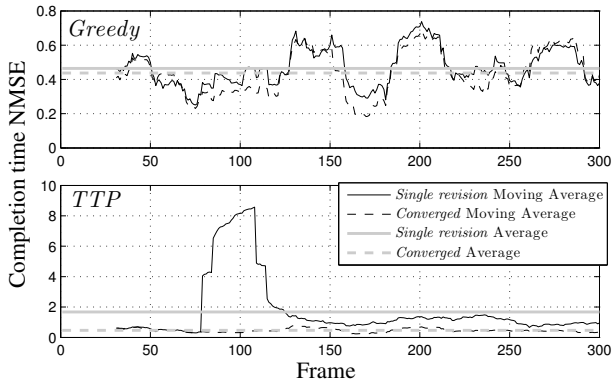

Fig. 3: Normalized completion time mean square error (NMSE) for *TTP* and *Greedy*. The figure shows the averages computed over all the frames and over a moving window of 30 frames for both *converged* and *single revision* updates.

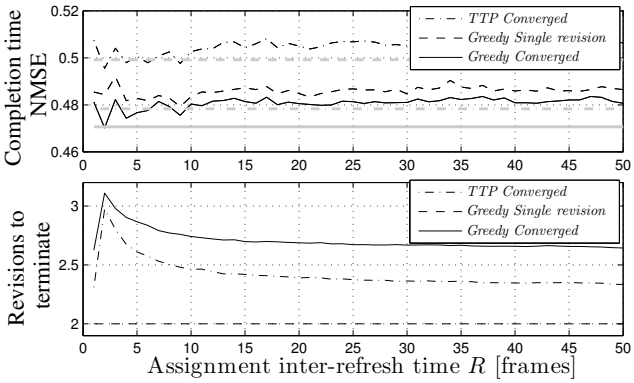

Fig. 4: Completion time NMSE (above) and number of iterations required to terminate (below) as a function of the assignment inter-refresh time R for *TTP* and *Greedy*. The results show the average over 20 runs.

while the processing time coefficients are such that a single image can be processed in $8 \cdot 10^{-2}$ seconds.

We evaluate the performance of the algorithms by computing the mean square error e^C of the system completion time $T^i = \max_s T_s^i$ compared to the optimal completion time $T^{i,*} = \max_s T_s^{i,*}$, defined as

$$e^C = \frac{1}{|\mathcal{I}_s|} \sum_{i=1}^{|\mathcal{I}_s|} \left(T^i - T^{i,*}\right)^2,$$

where $T_s^{i,*}$ is the completion time of sensor node s under the optimal allocation. In the figures we show the MSE of the considered algorithms normalized by the MSE of a non-adaptive algorithm where each sensor always uses the *fixed* assignment function and cut-point location combination that minimizes the MSE.

Throughout the section, we compare *TTP* to *Greedy* proposed in [11]. *Greedy* differs from *TTP* in that each sensor $s \in D_i$ *always* revises its assignment from (x_s^i, d_s^i) to $(x_s^{i+1}, d_s^{i+1}) = (x'_s, d'_s)$. Observe that Lemma 1 does not hold for *Greedy*, there is thus no guarantee that the system would reach a stable allocation in a finite number of revisions when using *Greedy*.

5.1. Impact of Convergence

Recall that *TTP* was defined to allow a *single revision* per image i. However, the sensors could perform repeated allocation revisions for the same image i. According to Theorem 1 this would allow them to compute a stable allocation $(\bar{\mathbf{x}}, \bar{\mathbf{d}})$. In the following we investigate whether implementing a stable allocation upon every image improves the system performance, and how it affects the computational complexity.

For the evaluation, we let the sensors perform at most 50 revisions upon each image i and we denote by $(\mathbf{x}^{i,k}, \mathbf{d}^{i,k})$ the allocation after the k-th revision. The allocation $(\mathbf{x}^{\mathbf{i}}, \mathbf{d}^{\mathbf{i}})$ implemented upon image i is either a stable allocation $(\bar{\mathbf{x}}, \bar{\mathbf{d}})$, if *TTP* terminates before 50 revisions, or the allocation $(\mathbf{x}^{i,50}, \mathbf{d}^{i,50})$ reached after 50 revisions. We refer to the allocation as *converged*. We compare the performance of *converged* updates with the case when the sensors perform a *single revision* as defined by *TTP* in Section 4. Observe that performing *converged* updates increases the computational load of the sensors, as each update requires each sensor $s \in D_{i,k}$ to compute the allocation (x'_s, d'_s) that minimizes its completion time.

Table 1: Average and standard deviation over all frames of the number of iterations needed by the sensors to reach a stable allocation and convergence ratio r, for *converged* updates.

		un-coord. assign.			*coord. assign.* ($R = 5$)		
Rev.	Alg.	avg.	std.d.	r	avg.	std.d.	r
Async.	*TTP*	2.71	0.87	1.0	2.63	0.73	1.0
	Greedy	3.19	0.95	1.0	2.84	0.92	1.0
Sync.	*TTP*	3.44	1.98	0.52	1.75	1.42	0.89
	Greedy	3.82	2.58	0.50	2.17	1.42	0.90

In Figure 3, we show the normalized MSE (NMSE) for *Greedy* and *TTP* under *asynchronous revision*, for the *single revision* and *converged*. First, we observe that letting the sensors reach a stable allocation provides on average a lower completion time, both for *Greedy* and *TTP*. Second, we observe that the difference between *single revision* and *converged* is significantly higher for *TTP* than for *Greedy*. This is due to that *TTP* only allows sensors to make a subset of the updates allowed by *Greedy*. In Table 1 we show the average and the standard deviation over all frames of the number of iterations needed by the sensors to reach a stable allocation, for all the simulation scenarios that we consider. We also show the ratio r of images for which the algorithm converges within 50 updates, versus the total number of images. Observe that $r = 1.0$ for *asynchronous revision*, both for *TTP* and *Greedy*, which shows that *Greedy* always reaches a stable allocation in practice. Furthermore, the results confirm that *TTP* requires less iterations to converge to a stable allocation, hence it may represent a good trade-off between computational complexity and performance.

5.2. Coordinated assignment mode

In the following we investigate the benefit of coordinated assignment in terms of completion time and convergence time.

In Figure 4 we show the completion time NMSE and the number of iterations required to terminate as a function of the allocation inter-refresh time R, for *TTP* and *Greedy*. The average completion time NMSE for the scenarios in Figure 3 is shown for comparison using gray lines. We observe that, if sensors can converge to a stable cutpoint location vector, the

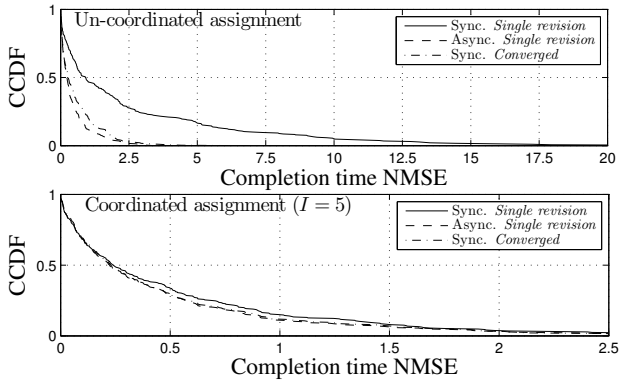

Fig. 5: Complementary CDF of the completion time NMSE for *synchronous revision* and *asynchronous revision* for the *un-coordinated assignment* (above) and the *coordinated assignment* (below) operation modes. The curves show the results for *Greedy* for *single revision* and *converged* updates.

completion time NMSE significantly increases in the coordinated assignment operation mode. For *single revision* updates instead, it pays off to coordinate the assignment, as one single revision is not sufficient for the system to reach a good allocation in the *un-coordinated assignment* operation mode. For both *converged* and *single revision* updates it is beneficial to update the assignment often, as for high values of R the provided assignment becomes obsolete and does not help to improve the performance of the system. The number of iterations needed to converge to a stable allocation decreases as the inter-refresh time R increases, and it is lower for *TTP*, which validates the analytical results in Section 4.

5.3. Impact of Revision Opportunity

We finally compare synchronous and asynchronous revisions. While synchronous revisions may allow the nodes to converge faster, convergence cannot be guaranteed and the resulting performance is unclear.

In Figure 5 we show the complementary CDF of the completion time NMSE for *synchronous revision* and for *asynchronous revision*. The figure confirms that *synchronous revision* may degrade the system performance in both *un-coordinated* and *coordinated assignment* modes. The completion time is significantly higher under *synchronous revision* for the *un-coordinated assignment mode*, as every sensor s is free to update its assignment vector d_s^i upon the same image i, sensors may select their allocations in a way that the load on certain processing nodes becomes very large. The completion time MSE is similar under *synchronous revision* and *asynchronous revision* for *converged* updates, although *Greedy* does not terminate for a significant fraction of the images, as shown in Table 1.

6. CONCLUSION AND FUTURE WORK

We considered the problem of completion time minimization in VSNs that include dedicated processing nodes for distributed visual analysis. We proposed a distributed algorithm for feature computation off-loading and showed that it converges to a stable off-loading assignment. Simulations performed on a surveillance video trace showed that the proposed algorithm converges faster than existing algorithms at the price of a slightly higher completion time. In case of strong computational constraints at the sensor nodes, the system can benefit from coordinating the operation of the sensor nodes through a central entity, provided that coordination is performed reasonably often.

7. REFERENCES

[1] A. Redondi, L. Baroffio, A. Canclini, M. Cesana, and M. Tagliasacchi, "A visual sensor network for object recognition: Testbed realization," in *Proc. Int. Conf. Digital Signal Process. (DSP)*, 2013.

[2] E. Eriksson, G. Dán, and V. Fodor, "Real-time distributed visual feature extraction from video in sensor networks," in *Proc. IEEE Int. Conf. Distributed Computing in Sensor Syst. (DCOSS)*, 2014.

[3] L. Baroffio, A. Canclini, M. Cesana, A. Redondi, M. Tagliasacchi, G. Dán, E. Eriksson, V. Fodor, J. Ascenso, and P. Monteiro, "Enabling visual analaysis in wireless sensor networks," in *Proc. IEEE Intl. Conf. Image Process. (ICIP), Show and Tell*, October 2014.

[4] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 105–119, 2010.

[5] H. Zhou, Y. Yuan, and C. Shi, "Object tracking using sift features and mean shift," *Comput. Vision and Image Understanding*, vol. 113, no. 3, pp. 345–352, 2009.

[6] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (SURF)," *Comput. Vision and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008.

[7] S. Leutenegger, M. Chli, and R. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, 2011.

[8] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Proc. European Conf. Comput. Vision (ECCV)*, 2010.

[9] G. Dán, M. A. Khan, and V. Fodor, "Characterization of SURF and BRISK interest point distribution for distributed feature extraction in visual sensor networks," *IEEE Trans. Multimedia*, vol. 17, no. 5, May 2015.

[10] V. Bharadwaj, D. Ghose, and T. Robertazzi, "Divisible load theory: A new paradigm for load scheduling in distributed systems," *Cluster Computing*, vol. 6, no. 1, pp. 7–17, 2003.

[11] E. Eriksson, G. Dán, and V. Fodor, "Algorithms for distributed feature extraction in multi-camera visual sensor networks," in *Proc. IFIP Networking*, May 2015.

[12] M. A. Khan, G. Dán, and V. Fodor, "Characterization of SURF interest point distribution for visual processing in sensor networks," in *Proc. Int. Conf. Digital Signal Process. (DSP)*, 2013.

[13] C. Tang and P. K. McKinley, "Modeling multicast packet losses in wireless lans," in *Proc. ACM Int. Workshop Modeling Anal. and Simulation of Wireless and Mobile Syst.*, 2003.

[14] J. Lacan and T. Perennou, "Evaluation of error control mechanisms for 802.11b multicast transmissions," in *Proc. Int. Symp. Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, 2006.

[15] J. Hartwell and A. Fapojuwo, "Modeling and characterization of frame loss process in IEEE 802.11 wireless local area networks," in *Proc. IEEE Veh. Technol. Conf. (VTC-Fall)*, 2004.

[16] R. Guha and S. Sarkar, "Characterizing temporal SNR variation in 802.11 networks," *IEEE Trans. Veh. Technol.*, vol. 57, no. 4, pp. 2002–2013, 2008.

[17] M. Petrova, J. Riihijarvi, P. Mahonen, and S. Labella, "Performance study of ieee 802.15.4 using measurements and simulations," in *Proc. IEEE Wireless Commun. and Networking Conf. (WCNC)*, 2006.

[18] T. Joshi, A. Mukherjee, Y. Younghwan, and D. Agrawal, "Airtime fairness for IEEE 802.11 multirate networks," *IEEE Trans. Mob.Comp.*, Apr. 2008.

[19] A. Kostuch, K. Gierlowski, and J. Wozniak, "Performance analysis of multicast video streaming in ieee 802.11 b/g/n testbed environment," in *Wireless and Mobile Networking*, ser. IFIP Advances in Information and Communication Technology, J. Wozniak, J. Konorski, R. Katulski, and A. Pach, Eds. Springer, 2009, vol. 308, pp. 92–105.