Ad Hoc Networks 28 (2015) 38–51

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Cooperative image analysis in visual sensor networks

Alessandro Redondi [a,*], Matteo Cesana [a], Marco Tagliasacchi [a], Ilario Filippini [a], György Dán [b], Viktoria Fodor [b]

[a] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, P.zza Leonardo da Vinci, 32, Milano, Italy
[b] Lab for Communication Networks, School of Electrical Engineering, KTH Royal Institute of Technology, Stockholm, Sweden

ARTICLE INFO

Article history:
Received 16 July 2014
Received in revised form 28 October 2014
Accepted 19 January 2015
Available online 30 January 2015

Keywords:
Visual sensor network
Distributed processing
Divisible load theory

ABSTRACT

This work addresses the problem of enabling resource-constrained sensor nodes to perform visual analysis tasks. The focus is on visual analysis tasks that require the extraction of local visual features, which form a succinct and distinctive representation of the visual content of still images or videos. The extracted features are then matched against a feature data set to support applications such as object recognition, face recognition and image retrieval. Motivated by the fact that the processing burden imposed by common algorithms for feature extraction may be prohibitive for a single, resource-constrained sensor node, this paper proposes cooperative schemes to minimize the processing time of the feature extraction algorithms by offloading the visual processing task to neighboring sensor nodes. The optimal offloading strategy is formally characterized under different networking and communication paradigms. The performance of the proposed offloading schemes is evaluated using simulations and is validated through experiments carried out on a real wireless sensor network testbed. The results show that the proposed offloading schemes allow to reduce the feature extraction time up to a factor of 3 in the reference scenario.

1. Introduction

Visual sensor networks (VSNs) extend the application fields of traditional wireless sensor networks by adding the capability to acquire and process multimedia signals such as still images and video. VSNs can have a significant impact in scenarios in which visual analysis is currently infeasible, due to the mismatch between the transmission and computational resources and the complexity of the analysis tasks. As an example, in the context of smart cities, the availability of inexpensive visual nodes can enable a much more complete coverage of the urban landscape, reaching a wider area and limiting the costs of the required infrastructure to support applications for traffic monitoring, smart parking metering, environmental monitoring, hazardous situations monitoring, etc. [1,2].

Classical networked systems for visual analysis follow the *compress-then-analyze* paradigm, where image/video analysis is performed last and is decoupled from the acquisition, compression and transmission phases. In the case of VSNs, powerful smart cameras are substituted by vision-enabled, battery-operated sensing nodes with low-power microprocessors and radio chips. The traditional compress-send-then-analyze paradigm may not fit well the computation and communication-related constraints imposed by these VSNs, as it requires large communication resources. For this reason, an alternative paradigm, named *analyze-then-compress* has been recently proposed leveraging the idea that most visual analysis tasks can be carried out based only on a succinct representation of the image [3]; that is, image features can be collected by sensing nodes, processed, and then delivered to the final

* Corresponding author. Tel.: +39 02 2399 9614; fax: +39 02 2399 3413.
E-mail address: redondi@elet.polimi.it (A. Redondi).

http://dx.doi.org/10.1016/j.adhoc.2015.01.008
1570-8705/

destination(s), thus avoiding the need for encoding and transmitting redundant pixel-level representations.

Extracting features from visual data is however a computationally intensive task. For the case of local features, the process entails detecting image keypoints and computing the corresponding descriptors, a process whose computational complexity grows linearly with the image size and with the number of scales, that is, the downsampled versions of the image that are used in the feature extraction process. As an example, even when using feature extraction algorithms tailored to low-power architectures (e.g. BRISK [4]), the processing time for detecting keypoints and generating the corresponding descriptors can be as high as 5*s* on an Intel Imote2 sensor platform [5], and in the order of 0.5–1 s when using a BeagleBone [6] based sensor node, as illustrated in Fig. 2.

Such processing time may not be low enough when the outcome of the visual processing task is used to trigger control actions or to carry out analysis tasks such as event detection and object tracking [7,8]. This observation calls for alternative computing strategies, that are able to reduce the processing time of a single frame in the VSN.

Different from classical networks of smart cameras where the camera itself is a stand-alone entity which does all the required processing, in the VSN scenario the camera node is most likely to be close to other wireless nodes, equipped with cameras or other kind of sensors. Such nodes may be responsible for the acquisition and processing of both visual and non-visual data (e.g., temperature, pressure, infrared radiation, etc.), and are characterized by specific processing and transmission capabilities. In such a scenario, proximity can potentially allow a camera node to leverage the resources of the neighboring sensor nodes to reduce the overall processing time of the visual processing task, following a distributed computing approach [1]. The main rationale is to let the camera node "borrow" processing power from the neighboring nodes to process part of the initial load (image), further exploiting the parallelization of the visual processing task. Ideally, parallelizing a processing task always leads to a lower processing time. However, in VSNs parallelization requires additional communication overhead, and therefore its optimization is not straightforward.

In this work, we target the minimization of the processing time, taking into account the limitations of the computational and communication capabilities of the sensors nodes and the specific requirements of the visual feature extraction. To optimize the distributed processing, we need to answer the following questions: (i) what communication paradigm, unicast or broadcast, is best suited for distributing the loads among the cooperators in VSNs, (ii) how many cooperators should be utilized and what is the load share each cooperator should get, (iii) what is the trade-off between processing time and energy consumption involved in the offloading process, and finally, and (iv) what is the impact of different visual contents on the overall processing time.

We formulate the problem of minimizing the processing time in the framework of Divisible Load Theory (DLT), which has been widely used to study how processing load can be optimally divided within processor grids [9]. We derive closed form expressions for the optimal task processing time and for the corresponding load assignment to the cooperating sensor nodes under different networking scenarios and communication paradigms, taking into consideration the specific properties of feature extraction algorithms. The performance of the proposed load distribution schemes are evaluated both through simulation and through the development of a real-life Visual Sensor Network composed of one camera node and several cooperators.

The rest of the paper is organized as follows. Section 2 discusses the related work further providing background on the divisible load theory. In Section 3 the main ideas and algorithms behind visual feature extraction are described; Section 4 defines the reference scenario and the problem statement. In Section 5 the DLT-based offloading framework for visual sensor networks is presented, and Section 6 reports the performance evaluation study of the proposed framework in realistic visual sensor networks. Section 7 concludes the paper.

2. Related work

The challenge of networked visual analysis under the *compress-then-analyze* paradigm is to minimize the amount of pixel data to be transmitted. As lossy coding in the pixel domain affects the visual analysis at the central node, recent works propose image coding schemes that are optimized for feature extraction [10,11].

If the *analyze-then-compress* approach is followed, the main challenge is to minimize the computational load at the camera node, and therefore many of the emerging feature extraction schemes aim at decreasing the computational complexity [12,4]. If the transmission bandwidth of the VSN is very limited, the objective can even be to limit the amount of data to be transmitted by compressing the descriptors or by limiting their number. Descriptor compression techniques are suggested and evaluated in [13,14]. In [15] a progressive transmission scheme is proposed, which terminates the transmission of new descriptors as soon as the image is retrieved. In [16] the number of considered interest points and the quantization level of the descriptors are jointly optimized to maximize the accuracy of the recognition, subject to energy and bandwidth constraints.

The present work is motivated by recent results on the expected transmission and processing load of visual analysis in sensor networks [5], demonstrating that processing at the camera or at the sink node of the VSN leads to significant delays, and thus distributed processing is necessary for real-time applications. To analyze the performance of distributed processing we leverage tools from the divisible load theory (DLT), which addresses the problem of load distribution in a grid of processors. DLT has been applied for different distributed systems [9], and recently also to wireless sensor networks [17,18], for processing large data whose computation may be split among different entities (i.e., in case the load is divisible).

Distributed processing in networks typically considers a central processor, which is the source of the processing

load, and a number of cooperating processors. The optimal load allocation, which minimizes the overall computational time, depends on the network architecture, that is, how many processors are available and how they are connected to the central processor, on the communication and computational capabilities of the processors, and on the relationship between the amount of allocated data to be processed and the processing time. Given this information, the optimal allocation can be computed through closed-form recursive equations, which characterize the offloading order and the load share to be assigned to each node. Several works derived such equations for different system scenarios: in [19] the load allocation is solved for bus-like networks, in [20] the solution is given for single-level tree (star) networks and in [21] the asymptotic performance analysis of linear and tree networks is presented. In this work we consider VSNs arranged in a star topology.

In [20], the three main properties of the optimal allocation are derived for the scenario when the central processor is not equipped with a front-end, that is, it cannot compute and communicate at the same time, and the processing time is linear in the amount of allocated data. First, under optimal load distribution all the processors stop computing at the same time. Second, the load distribution from the central processor is performed in decreasing order of link transmission capacities, independently from the processing speeds of the nodes. Third, a processor can be used only if the time needed to communicate a given load to the cooperating processor is lower than the time needed to process the same load the central processor. As we show, these general findings of DLT need to be adapted and extended to the specific network scenario and application requirements to be applicable to VSNs.

3. Background on feature extraction

Visual features provide a concise, yet efficient, representation of the underlying pixel domain content of an image, and are robust to many global and local transformations. Hence, they are employed for various visual analysis tasks, including image/video retrieval, object recognition, structure from motion, etc.

Extracting visual features from an image is accomplished with the aid of two components: the detector, which identifies salient keypoints in the image; and the descriptor, which provides a succinct representation of the image patch around each keypoint. Many algorithms have been recently proposed for these two components: detectors can identify blob-like [22,23] or corner-like [12,24] structures in the image and may or may not be scale/viewpoint/illumination invariant. In particular, using scale-invariant detectors, keypoints are detected at different *scales*, each scale being a resized and smoothed version of the original image. Similarly, several descriptor algorithms are available, ranging from local-gradient approaches [22,23], to faster methods based on pixel intensity comparisons [25,4].

The process of visual feature detection and description is the same regardless of the particular detector/descriptor algorithm used, and is shown in Fig. 1. It is important to note that to compute a visual feature at a certain location, a patch of pixels surrounding the location is required.[1] Moreover, the size of such a patch of pixels is not fixed, but depends on the particular scale at which the keypoint is found. In general, the higher the scale, the bigger the size of the patch needed to compute the descriptor. We will explain later why this aspect is critical in the design of the load distribution scheme.

In this work, we focus on the recently proposed BRISK [4] feature extraction algorithm. In BRISK, a fast multi-scale corner detector identifies salient keypoints in an image, which are then described efficiently starting simply from intensity comparisons between pixels of the patch to be described. As shown in several works [26,27], BRISK constitutes a good alternative to state-of-the-art methods such as SIFT [22] and SURF [23], providing similar performance at relatively lower computational times. BRISK is thus a good candidate for use in resource constrained architectures such as VSNs. Despite its relatively low computational complexity, running BRISK on commercially available visual sensor nodes can still result in excessive processing times. As an example, Fig. 2 shows the execution time of both the keypoint detection and the feature description of BRISK, when executed on a BeagleBone-based visual sensor node and using a VGA (640x480) image as input. As one can see, the time taken for detecting and extracting features has a minimum value of 300 ms and grows linearly with the number of keypoints detected in the input image. Typically, for visual analysis tasks such as object recognition or tracking, hundreds or even thousands of keypoints are required. Thus the time to extract the features may easily approach values in the order of one second. The delay values can even be higher under increased image resolution or when the sensors need to apply some energy saving policy, such as dynamic frequency scaling (DFS). Such latencies may critically impair the correct operation of the VSN, especially when it is required to react to detected events by means of control actions (e.g., tracking the recognized object, increasing the frame rate, zooming on a particular area). Minimizing the processing time for analyzing a frame is thus of crucial importance in VSNs.

4. System model and problem statement

We consider a VSN consisting of one camera node (the central processor) and a set $\mathcal{N}$ of cooperating nodes, as shown in Fig. 3. Without loss of generality, we assume that the nodes are indexed from 0 (the camera node) to $N = |\mathcal{N}|$. The camera node acquires an image of size L bits, and has to perform visual analysis by detecting and extracting visual features from it. The camera node can perform the processing itself, but it can also distribute parts of the processing to a subset $\mathcal{N}_u \subseteq \mathcal{N}$ of the cooperating nodes. We denote the number of cooperating nodes that are actually used by $n = |\mathcal{N}_u|$, and without loss of generality we assume that whenever $n < N$, the used

[1] In general, a patch of pixels is required also for the detection process. As an example, most of the recent corner detectors are based on FAST, which identifies a pixel p as a keypoint if n contiguous pixels in the Bresenham circle of radius 3 around p are brighter or darker than p.

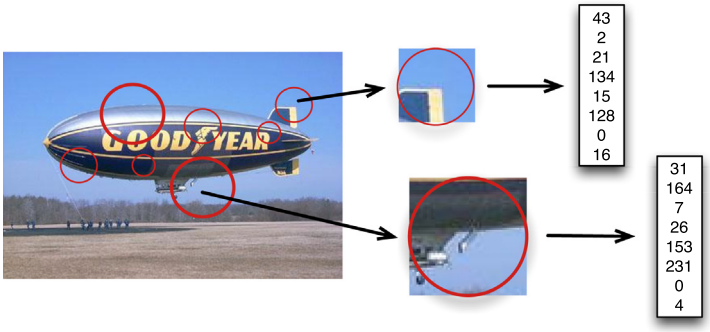


Fig. 1. Visual features extraction: several keypoints (red circles) are detected on the input image. The patch around each keypoint is then encoded in a numerical vector by the descriptor algorithm. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

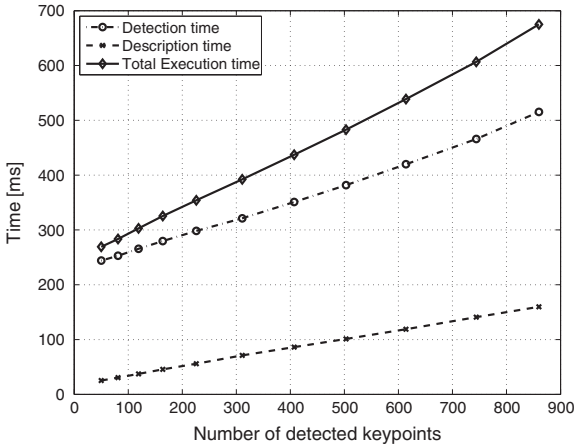


Fig. 2. BRISK execution time on a BeagleBone Linux computer and a VGA image.

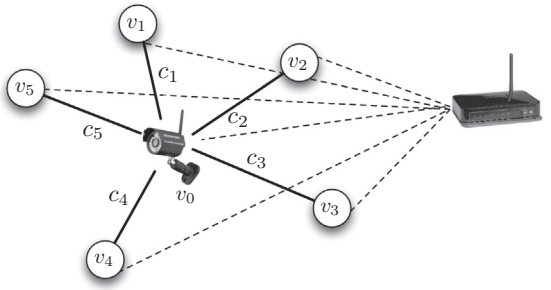


Fig. 3. A visual sensor network composed of one camera and 5 cooperating nodes. Each node is characterized by its processing speed v_i and its effective transmission rare c_i. After extraction, visual features are transmitted to a central controller (sink node) through links with equal capacity (represented with the dashed lines).

cooperating nodes are assigned indexes 1 to n. $n = 0$ corresponds to not using any cooperating node.

For distributing the load, the pixel-level representation of the image is split into n vertical slices.[2] All slices have the same height but can have different widths. We denote the width of slice i by α_i, where $0 \leqslant \alpha_i \leqslant 1$ is normalized with respect to the image's width and $\sum_{i=0}^{N} \alpha_i = 1$. By definition $\alpha_i = 0$ for $n < i \leqslant N$. The size of slice i in bits is $\alpha_i L$, and we refer to $\alpha = (\alpha_0, \ldots, \alpha_N)$ as the load assignment schedule. Each slice is then assigned and transmitted for processing (keypoint detection and descriptor extraction) to a cooperating node. We consider two scenarios for slice transmission to the cooperators: (i) slices are transmitted in raw pixel format or (ii) slices are compressed with a visual features preserving JPEG algorithm before transmission [11,28,29]. When transmitting in raw pixel format, there is a direct correspondence between the number of pixels in a slice and the size of the slice in bits. For the case of JPEG compression, we denote by γ the JPEG compression ratio, and thus the number of bits for slice i becomes $\alpha_i L/\gamma$. Experimental results in [28] show that by using $\gamma = 4$ all features are preserved.

[2] Note that the modeling framework holds if the split is performed in horizontal slices.

4.1. Node processing and communications capabilities

Each node is characterized by its communication and computational capabilities. Node i is characterized by its speed of processing one information bit, v_i (in bit/s) (for $0 \leqslant i \leqslant N$). We consider that the computational time spent by the i-th processor to detect keypoints and to extract features from the assigned image slice depends only on the size of the image slice itself, i.e., $T_{\mathrm{CPU},i} \propto \alpha_i$. In general, the time needed for extracting features depends both on the image size *and* on the visual image content through the number of features that need to be extracted [27]. Nevertheless, the assumption of proportionality holds if the keypoints in the input image are uniformly distributed, which was found to hold on average both horizontally and vertically in recent work [30]. We will evaluate the effect of non-uniformly distributed keypoints on the overall performance via experiments.

The sensor nodes use a wireless link layer technology capable of unicast and brodcast transmission, such as IEEE 802.15.4 or 802.11. In case of unicast offloading, the camera node sends the load share to each processing node

i via unicast transmissions. We assume that frames lost due to the impairments of the wireless channel are retransmitted in the link layer. Therefore, we characterize the transmission link by τ_i, the average time needed to transmit successfully one bit, and parameterise it with the effective link transmission rate c_i (in bit/s), such that $\tau_i = \frac{1}{c_i}$. In case of broadcast offloading the camera node uses broadcast transmission to transmit the part of the image that will be processed by the cooperating nodes. As link layer acknowledgements and retransmissions are not possible in typical technologies such as IEEE 802.15.4 and 802.11, we characterize the broadcast link with the raw transmission rate c_{bc}. The broadcast transmission rate is limited by the modulation and coding scheme that ensures correct decoding of the frames at the processing node with the weakest link, that is, $c_{bc} \leqslant \min_i c_i$. We denote by p_i the frame loss rate experienced by node i when using broadcast transmission. After the broadcast transmission, each processing node informs the camera node of the lost packets it needs to be retransmitted. The camera node then uses unicast transmission to retransmit the lost packets.

For link layer technologies with small frame sizes, such as IEEE 802.15.4, the transmission overhead may be significant. To capture this overhead, we denote by F the frame size in bits and by H the size in bits of the frame header (so that the effective payload is $F - H$ bits), and we compute the overhead as $h = \frac{F}{F-H}$.

The extracted features are finally sent to a central controller (sink node) for further processing (e.g., object recognition/tracking). We do not model the time needed for transmitting the extracted features to the central controller, as this delay appears in the visual system regardless of whether the processing is centralized at the camera or distributed among the sensor nodes.

To model the energy consumption of the sensor nodes, we denote by P_{cpu} the processing power of the sensor nodes, and we denote by P_{tx} and P_{rx} the transmit and the receive power of the sensor nodes, respectively. The energy use is proportional to the time spent processing, transmitting and receiving.

4.2. Overlap between slices

Offloading must not negatively affect the accuracy of visual analysis, that is, the overall set of keypoints extracted by the camera and by the cooperating nodes must be the same as the set of keypoints that would be extracted by the camera node alone. Recall that each feature descriptor is computed based on a patch of pixels around the corresponding keypoint. The size of the patch depends on the scale at which the keypoint is detected, small patches are needed at lower scales and larger ones at higher scales. Thus, in order to avoid that keypoints on the boundary between two adjacent slices get lost, as shown in Fig. 4, the input image has to be split in overlapping slices. More in details, the overlap is required to avoid errors in the computation of the features detected close to the border of each slice. The amount of the overlap is proportional to the maximum size of the patch required by the particular descriptor algorithm. We denote by o (in percentage with respect to the total image size L) the overlap between two adjacent slices. Fig. 4 illustrates the overlap between slices. For convenience we will consider that the overlap between two slices is attached to the slice with lower index. Note that this implies that the last slice of the image does not need any overlap. An equivalent notation would be to attach a smaller $(o/2)$ overlap on both sides of each slice. To capture the overlap associated to each slice, we define $\beta_i = \alpha_i + o$ for $i < n, \beta_n = \alpha_n$ and $\beta_i = 0$ for $n < i \leqslant N$. Thus, the amount of data corresponding to the i-th slice including the overlap is $\beta_i L$. Observe that if $o > 0$ and $n > 0$ then $\sum_{i=0}^{N} \beta_i > 1$.

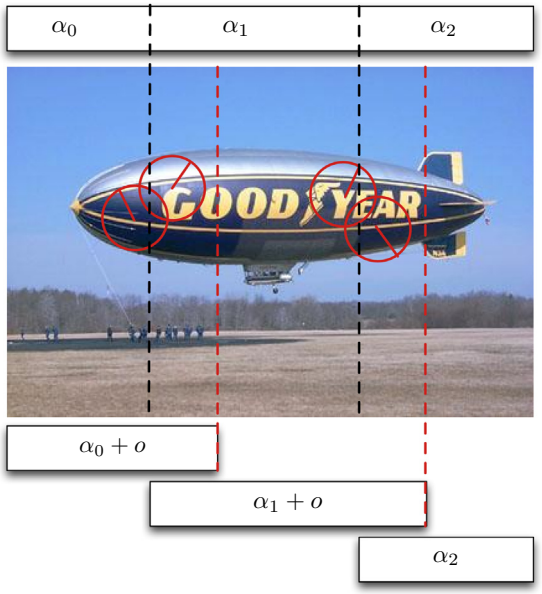


Fig. 4. Image splitting with vertical cuts and overlap.

The overlap has to be chosen such that features at the highest scale (those requiring the largest patch of pixels) are correctly computed. Formally, if we denote by $\mathcal{P}$ the set of keypoints detected in the whole input image, and by $\mathcal{P}_i$ the sets of keypoints detected in the i-th slice of the same image for an overlap o, then we would like $\mathcal{P} \equiv \bigcup_i \mathcal{P}_i$.

We performed experiments to assess the amount of overlap needed for correct descriptor extraction under slicing. To quantify the similarity between two sets of keypoints we used the *repeatability* measure. Repeatability is a widely used measure to evaluate detectors, and was originally defined in [31] as the ratio between the number of point-to-point correspondences and the minimum number of keypoints detected in a given pair of images. In our case, we define the repeatability measure as follows

$$Repeatability = \frac{|\mathcal{P} \cap \bigcup_i \mathcal{P}_i|}{|\mathcal{P}|} \tag{1}$$

Hence, the repeatability is 100% if and only if $\mathcal{P} \equiv \bigcup_i \mathcal{P}_i$.

Fig. 5 shows the repeatability measure for different values of the overlap, when using the BRISK detector with standard parameters (4 octaves and a detection threshold

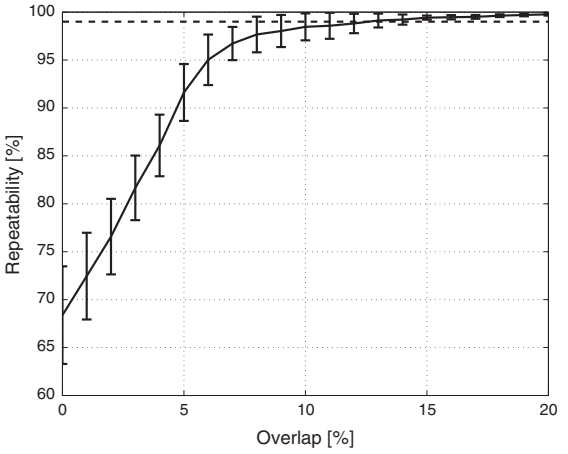


Fig. 5. Impact of the splitting overlap on the repeatability of keypoints.

of 60) and averaged over 50 different input VGA images (640×480). The results show that an overlap of about 15% of the total image size gives a repeatability score above 99%, and confirm that slicing with an appropriate overlap ensures correct keypoint detection and feature extraction. Experiments with different detector parameters led to similar results, and are omitted here for brevity.

4.3. Problem statement

The problem we address is the design of the optimal offloading scheme that minimizes the completion time for the task of visual feature extraction. We define the total completion time as $T = \max_{0 \leqslant i \leqslant n} T_i$, where T_i is the time instant when node i completes its assigned task.[3] To find the minimum completion time T we need to determine (i) the best offloading mode (unicast or broadcast), (ii) the number n of cooperating nodes to be used, (iii) the load share α_i assigned to the cooperating nodes and (iv) the order in which the loads should be assigned to the n cooperating nodes. The solution that minimizes the total completion time T depends on the network configuration, that is, on the processing and transmission capabilities $\nu_i, c_i, c_{\text{bc}}, p_i$ (with $0 \leqslant i \leqslant N$) and on the required overlap o.

Besides the completion time, we are also interested in the impact of offloading on the VSN lifetime $\mathcal{T}_{\text{VSN}}$. We define the VSN lifetime $\mathcal{T}_{\text{VSN}}$ as the total number of images that can be analyzed by the network before the energy of the camera or one of the processing nodes is depleted. Without loss of generality, we assume that every sensor node has the same energy budget $\overline{E}$ so that the lifetime of the VSN can be expressed as

$$\mathcal{T}_{\text{VSN}} = \frac{\overline{E}}{max_i E_i}, \tag{2}$$

where E_i is the energy consumption of node i. Given the load assignment schedule α, and the energy consumption parameters $P_{\text{cpu}}, P_{\text{tx}}$ and P_{rx}, we can express E_i, the energy consumption of node i for unicast offloading as

$$E_i = \begin{cases} L\left[P_{\text{cpu}} \frac{\beta_0}{\nu_0} + P_{\text{tx}} h\left(\sum_{i=1}^{N} \frac{\beta_i}{\gamma c_i}\right)\right] & i = 0 \\ L\left[P_{\text{cpu}} \frac{\beta_i}{\nu_i} + P_{\text{rx}} h \frac{\beta_i}{\gamma c_i}\right] & 1 \leqslant i \leqslant N \end{cases} \tag{3}$$

The energy consumed by the camera node ($i = 0$) is the sum of the required energy for processing its own slice of the original image (first term) and of the required energy to transmit via unicast the remaining part of the original image to the cooperating nodes (second term). Each cooperating node ($1 \leqslant i \leqslant N$), similarly, consumes energy for receiving and processing the assigned slice.

In the case of broadcast offloading the energy consumption E_i is given by

$$E_i = \begin{cases} L\left[P_{\text{cpu}} \frac{\beta_0}{\nu_0} + P_{\text{tx}} h\left(\frac{(1-\alpha_0)}{\gamma c_{\text{bc}}} + \sum_{i=1}^{N} \frac{p_i \beta_i}{\gamma c_i}\right)\right] & i = 0 \\ L\left[P_{\text{cpu}} \frac{\beta_i}{\nu_i} + P_{\text{rx}} h\left(\frac{1-\alpha_0}{\gamma c_{\text{bc}}} + \frac{p_i \beta_i}{\gamma c_i}\right)\right] & 1 \leqslant i \leqslant N. \end{cases} \tag{4}$$

That is, the energy consumed by the camera node ($i = 0$) is the sum of the required energy for processing its own slice of the image (first term) and of the required energy for transmitting the remaining slices to the cooperating nodes (second term). The energy required for transmission consists of the energy required for transmitting in broadcast the remaining part of the image ($P_{\text{tx}} h \frac{(1-\alpha_0)}{\gamma c_{\text{bc}}}$) and of the energy required for re-transmitting in unicast the slices of the image to specific cooperators in case the broadcast transmission fails ($P_{\text{tx}} h \sum_{i=1}^{N} \frac{p_i \beta_i}{\gamma c_i}$). Similarly, the energy consumed by the cooperating nodes is the sum of the energy required for processing the assigned image slice (first term) and the energy for receiving the broadcast transmission and any unicast retransmissions, in case the broadcast transmission fails.

5. Optimal offloading for visual sensor networks

In the following we derive closed-form expressions for the optimal load assignment α^* in the considered VSN for the case of unicast and for the case of broadcast wireless transmission. First we consider that the set $\mathcal{N}_u$ of used cooperating nodes and the scheduling order are given. Since for $\mathcal{N}_u = \emptyset$ the optimal offloading schedule is trivial, we consider that $\mathcal{N}_u \neq \emptyset$. Then we discuss how to apply the properties of optimal allocation [20], discussed in Section 2, for our specific scenario.

5.1. Unicast offloading

In the case of unicast offloading the camera node transmits slices of the image to the set of used cooperating nodes via individual unicast transmissions. Fig. 6 shows the timing diagram for unicast offloading including the transmission time and the processing time at the cooperating nodes. Since the load is not perfectly divisible, each slice to be offloaded contains also the overlap (for all but the last cooperating node).

We first recall a basic result from divisible load theory [9].

[3] The starting point $t = 0$ is set to the time when the camera node has finished acquiring the image and the processing task can then start.

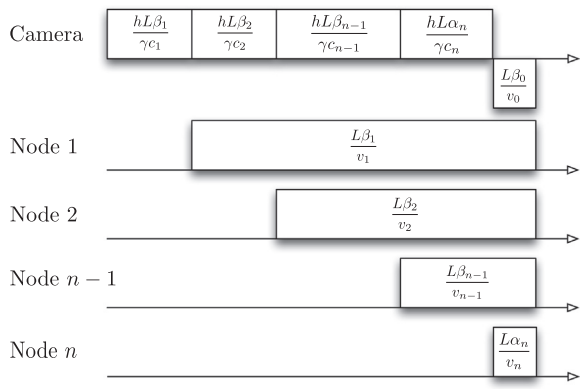


Fig. 6. Unicast splitting and offloading.

Lemma 1. *The optimal load assignment schedule α^* is such that the used cooperators and the camera finish the computation at the same time instant, that is, $T_i = T, 0 \leqslant i \leqslant n$.*

We can use the above insight to characterize the optimal load assignment schedule for a given set $\mathcal{N}_u$ of used cooperators.

Proposition 1. *The optimal load assignment schedule for unicast offloading using nodes in $\mathcal{N}_u \neq \emptyset$ is the solution to the following linear system*

$$\frac{\beta_0}{v_0} = \frac{\beta_n}{v_n} \tag{5}$$

$$\frac{\beta_i}{v_i} = \frac{\beta_{i+1}}{v_{i+1}} + h\frac{\beta_{i+1}}{\gamma c_{i+1}}, \quad 1 \leqslant i \leqslant n-1 \tag{6}$$

$$\beta_i = \alpha_i + o, \quad 0 \leqslant i \leqslant n-1 \tag{7}$$

$$\beta_n = \alpha_n \tag{8}$$

$$\sum_{i=0}^{N} \alpha_i = 1. \tag{9}$$

Proof. Let us consider Fig. 6. Observe that the camera and node n spend the same amount of time processing, hence (5). Next, the processing time of node $n-1$ equals the time to transmit data to node n and the processing time of node n, hence (6) for $i = n-1$. The other the equalities follow similarly. □

The solution to the above set of equations is the optimal load assignment schedule for a particular permutation of the nodes in $\mathcal{N}_u$, and is given in the following.

Corollary 1. *The optimal load assignment schedule under unicast offloading for a permutation of the set $\mathcal{N}_u$ of used nodes is given by*

$$\alpha_n^* = \beta_n^* = \frac{1 + on}{1 + \frac{v_0}{v_n} + \sum_{i=1}^{n-1} \prod_{j=i}^{n-1} v_j \left(\frac{1}{v_{j+1}} + \frac{h}{\gamma c_{j+1}}\right)}, \tag{10}$$

and (5)–(8). The completion time corresponding to the optimal load assignment schedule α^* is

$$T_{\text{uc}} = L\left(\frac{\beta_1^*}{\gamma c_1}h + \frac{\beta_1^*}{v_1}\right) = L\frac{\beta_n^*}{v_n\left(\frac{1}{v_1} + \frac{h}{\gamma c_1}\right)}\left(\frac{h}{\gamma c_1} + \frac{1}{v_1}\right). \tag{11}$$

Proof. Observe that (9) is equivalent to $\sum_{i=0}^{n} \beta_i = 1 + on$. The expression for β_n^* can then be obtained by recursively expressing β_i as a function of β_{i+1}. □

5.2. Broadcast offloading

In the case of broadcast offloading the camera node first broadcasts the part of the image that will be processed by the cooperating nodes, with rate c_{bc}. Due to the broadcast transmission, the overlaps do not need to be transmitted twice. After the broadcast transmission, missing frames from slices and overlaps are retransmitted via a sequence of unicast transmissions to each node i, now with link layer retransmissions, with effective rate c_i.

Fig. 7 shows the time diagram for broadcast offloading including the transmission times and the processing times of the cooperating nodes.

Again, the optimal load assignment schedule can be found by imposing that all used cooperating nodes complete computation at the same time.

Proposition 2. *The optimal load assignment schedule for broadcast offloading for a permutation of the set $\mathcal{N}_u$ of used nodes is the solution to the following linear system*

$$\frac{\beta_0}{v_0} = \frac{\alpha_n}{v_n} \tag{12}$$

$$\frac{\beta_i}{v_i} = \frac{\beta_{i+1}}{v_{i+1}} + p_{i+1}\frac{h\beta_{i+1}}{\gamma c_{i+1}}, \quad 1 \leqslant i \leqslant n-1 \tag{13}$$

$$\beta_i = \alpha_i + o, \quad 0 \leqslant i \leqslant n-1 \tag{14}$$

$$\beta_n = \alpha_n \tag{15}$$

$$\sum_{i=0}^{n} \alpha_i = 1. \tag{16}$$

Proof. Let us consider Fig. 7. Recall that p_i is the probability that the i-th cooperator loses a packet during broadcast transmission, and thus the amount of data to be transmitted in unicast to node i is $p_i(L\beta_i)$. Unicast transmission to node i thus takes $\frac{p_i(hL\beta_i)}{\gamma c_i}$ time. Node i can start processing its load once it receives the frames lost during broadcast. The equations are then obtained using a similar reasoning as in the proof of Proposition 1. □

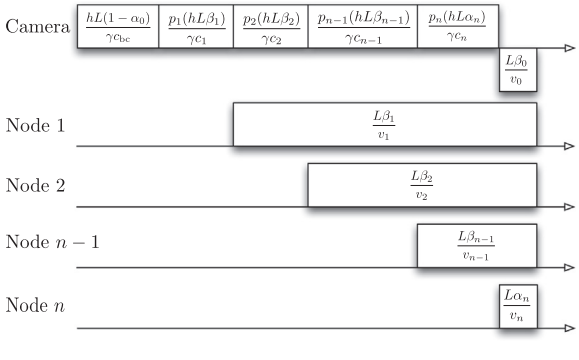


Fig. 7. Broadcast offloading.

The solution to the above set of equations, the optimal load assignment schedule under broadcast offloading, is given in the following.

Corollary 2. *The optimal load assignment schedule under broadcast offloading for a permutation of the set $\mathcal{N}_u$ of used nodes is given by*

$$\alpha_n^* = \beta_n^* = \frac{1+on}{1+\frac{v_0}{v_n}+\sum_{i=1}^{n-1}\prod_{j=i}^{n-1} v_j\left(\frac{1}{v_{j+1}}+\frac{p_{j+1}h}{\gamma c_{j+1}}\right)}, \tag{17}$$

and (12)–(15). The completion time corresponding to the optimal load assignment schedule α^* is

$$\begin{aligned} T_{\mathrm{bc}} &= L\left[\frac{h(1-\alpha_0)}{\gamma c_{\mathrm{bc}}}+\frac{p_1(h\beta_1)}{\gamma c_1}+\frac{\beta_1}{v_1}\right] \\ &= L\left[\frac{h(1+o)}{\gamma c_{\mathrm{bc}}}-\frac{hv_0\beta_n^*}{v_n\gamma c_{\mathrm{bc}}}+\frac{\beta_n^*}{v_n\left(\frac{1}{v_1}+\frac{h}{\gamma c_1}\right)}\left(\frac{p_1 h}{\gamma c_1}+\frac{1}{v_1}\right)\right]. \end{aligned} \tag{18}$$

5.3. Scheduling order and the optimal set of cooperating nodes

While the optimal scheduling order is in the decreasing order of link transmission rates in many basic scenarios, as derived in [20], in the presence of transmission overlap the optimal order depends also on the processing rates of the nodes [32], and no general results can be derived for networks with heterogeneous transmission and processing rates. Therefore, in the numerical evaluation we consider homogeneous processing rates, where the basic result on the optimal scheduling order holds.

We select the set of cooperating nodes as proposed in [20], comparing the time needed for processing information at the camera to that for transmitting it and processing it at a network node. Based on this principle we can formulate sufficient conditions for a potential cooperating node not to belong to the set of actual cooperating nodes.

Proposition 3. *Under unicast transmission, if cooperating nodes are added in decreasing order of effective transmission rate, an additional node i should not be added to the set $\mathcal{N}_u$ of used cooperating nodes if*

$$\frac{\alpha_i}{v_0} < \frac{\alpha_i}{\gamma c_i}+\frac{o}{\gamma c_i}. \tag{19}$$

Proof. According to [20], a processing node should be added only if processing the same load at the camera takes more time than the transmission time it introduces in the network. In the case of overlaps, adding one additional node also introduces transmission of one more overlap to the previous node, which gives the result in (19). □

Under broadcast transmission due to adding one more cooperating node the broadcast transmission rate c_{bc} may need to be decreased, with a consequence of changing (most probably decreasing) the loss probability p_i at all the cooperating nodes. In this case the effect of the addition of one more processing node needs to be evaluated numerically. For the specific case when c_{bc} does not need to be changed, the following holds.

Proposition 4. *Under broadcast transmission, if cooperating nodes are added in decreasing order of effective transmission rate, and the transmission rate c_{bc} does not need to be changed, an additional node i should not be added to the set $\mathcal{N}_u$ of used cooperating nodes if*

$$\frac{1}{v_0} < \frac{p_i}{\gamma c_i}. \tag{20}$$

Proof. In this case the processing of slice α_i at the camera node needs to be compared to the transmission time of the frames lost from the slice under the broadcast transmission which gives (20). □

6. Performance evaluation

In this section we evaluate the benefit of unicast and broadcast offloading in speeding up the processing of stand-alone images. For the numerical results we consider a VSN consisting of BeagleBone Linux computers [6], integrated with a 802.11 g compliant dongle to form a visual sensor node. The camera node is equipped with a low cost USB camera with a resolution of 640×480 pixels, 8 bits per pixel. We consider two scenarios for the link transmission rates. In the *homogeneous* scenario all links have the same effective transmission rate $c_i = 24$ Mbps. In the *heterogeneous* scenario the effective transmission rates are chosen uniform at random from the set of available data rates in 802.11 g (i.e., 6, 9, 12, 18, 24, 36, 48, 54 Mbps). For simplicity, for the broadcast offloading we set c_{bc} to the minimum effective transmission rate of the participating nodes. The results presented for this scenario are the averages of 100 simulations. We consider that the probability of losing a packet is equal on all links (i.e., $p_i = p$) and use values of 0, 5% and 20%, corresponding to an ideal, average and bad case, respectively. For simplicity, all nodes have the same processor speed (i.e., $v_i = v$). The BRISK algorithm is parametrized so that 500 keypoints are detected and the required overlap is 15% of the image size. The complete set of parameters used in our analysis is reported in Table 1.

Table 1
Parameters used for the analysis.

Name	Symbol	Value
Image load	L	2.46×10^6 bits
JPEG compression ratio [11]	γ	4
CPU speed	v	1.48 Mbps
CPU power	P_{cpu}	2.1 W
RADIO TX power	P_{tx}	1.5 W
RADIO RX power	P_{rx}	1.2 W
Energy budget	$\overline{E}$	32.4×10^3 J
Packet size	F	1512 Bytes
Header size	H	66 Bytes
Image overlap	o	15%

6.1. Simulation results

We first focus on the impact of the number of processing nodes on the completion time. We report the completion times normalized by the completion time in case the processing is entirely carried out by the camera node, referred to as a speedup ratio. For each figure we show two families of curves, depending on which assumption is made for the transmission of image slices to cooperators. We add the label "JPEG" to those curves which have been obtained by assuming that slices are compressed with a feature preserving JPEG encoding algorithm (with $\gamma = 4$) before transmission.

Fig. 8(a) and (b) show the speedup obtained by the two offloading schemes for the homogeneous and the heterogeneous scenarios, respectively with an increasing number of available nodes N. The results were obtained as follows. For a given value of N, the potential cooperators were first sorted in decreasing order of transmission capacity. Then, the list of potential cooperators was pruned using Propositions 3 or 4. Finally, the total completion time was calculated as a function of the number of used cooperators n. The optimal n is the one that maximizes the processing speed up. It is interesting to observe that broadcast offloading always outperforms unicast offloading in the homogeneous scenario, while the opposite holds for the heterogeneous case. Clearly, in the homogeneous scenario broadcasting eliminates the transmission redundancy introduced by the overlap at no loss of transmission rate, but in the heterogeneous scenario the relatively low c_{bc} increases the time to transmit the whole image compared to unicast. Compressing slices before transmission clearly allows to obtain higher speedups.

For the considered input parameters all the available nodes could be used in the homogeneous scenario, while in the heterogeneous case the ones with low transmission rate were not used.

Fig. 9 shows the speed-up vs. the number n of used cooperating nodes for $N = 10$, heterogeneous transmission capacities and uncompressed slices. Observe that the optimal number of used cooperating nodes is lower under broadcast (6 nodes) than under unicast (8 nodes). This is due to that the broadcast offloading scheme is more restrictive, as nodes with weak links limit c_{bc}.

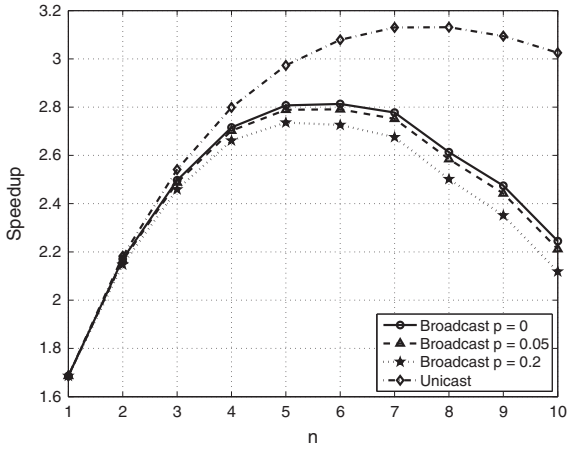
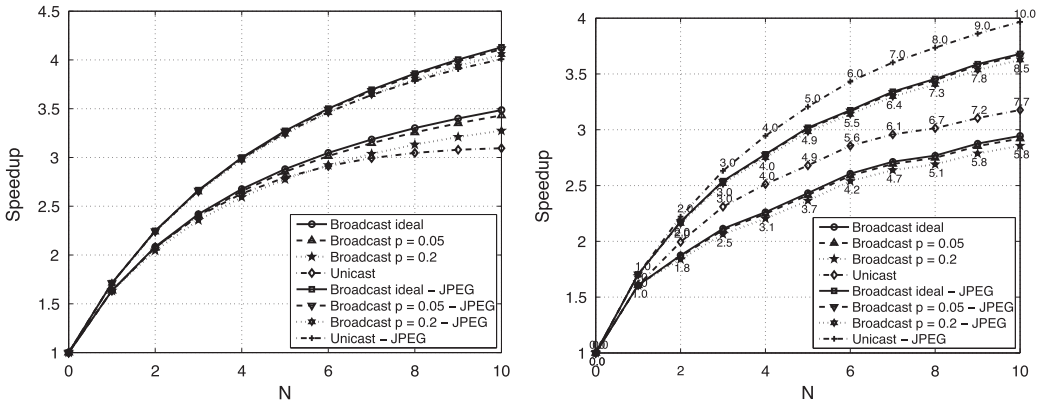


Fig. 9. Speedup vs. number of used cooperating nodes n when the number cooperating nodes is $N = 10$.

Fig. 10 shows the speedup for $N = 5$ cooperating nodes for the heterogeneous scenario as a function of the overlap o. The broadcast strategy suffers less from the increase of the overlap. Again, in absolute terms, the unicast strategy achieves the best performance.

Fig. 11(a) and (b) show the VSN lifetime obtained by the two offloading schemes for homogeneous and heterogeneous link transmission rates respectively. The energy budget of the nodes, $\overline{E}$ is given in Table 1. We observe the same trend as observed for the speed-ups; broadcast offloading is the best choice for homogeneous case, while unicast offloading is better for the heterogeneous one.

Finally, Fig. 12 shows the trade-off between energy consumption and speedup. We quantify the energy consumption by normalizing the total energy consumption with that of a non-cooperative VSN that consist of the camera node only ($N = 0$). Since the processing energy is the same for the non-cooperative and for the cooperative case (the entire image needs to be processed), the total energy



(a) Homogeneous scenario. (b) Heterogeneous scenario.

Fig. 8. Speedup vs. number of available cooperating nodes N.

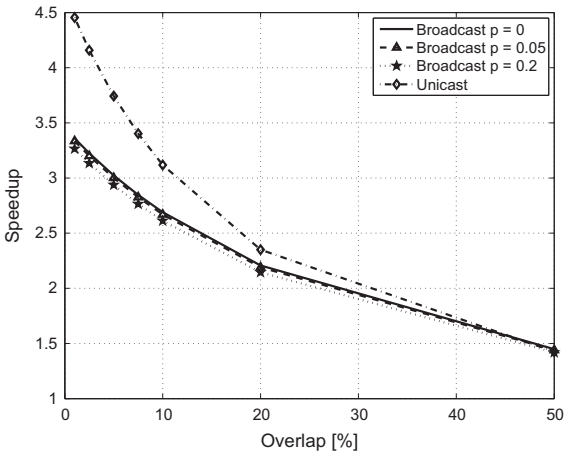


Fig. 10. Speedup vs. overlap for the heterogeneous scenario, $N = 5$.

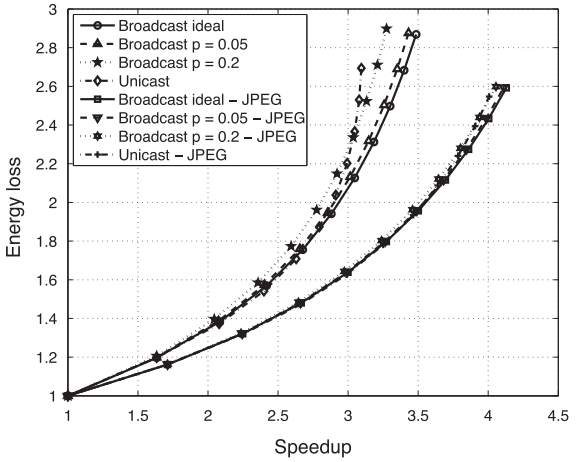


Fig. 12. Tradeoff between total energy consumption and speedup in case the number of available cooperating nodes ranges from 0 to $N = 10$.

consumption of the cooperative scheme is higher, as it includes transmission and reception costs. Therefore, we refer to the normalized energy consumption as *energy loss*. The results show that the speedup exceeds the energy loss over the considered range. As an example, the total energy required to achieve a speedup of 3 with the unicast strategy requires 2 times the energy needed by the camera alone in the non-cooperative case if slices are non compressed, whereas the energy loss is limited to 1.6 times in the case of compressed slices. However, the speed up has an upper limit, and close to this limit the marginal energy loss is very high. Consequently, an energy-optimized cooperative system should only aim for a speedup required by the visual application.

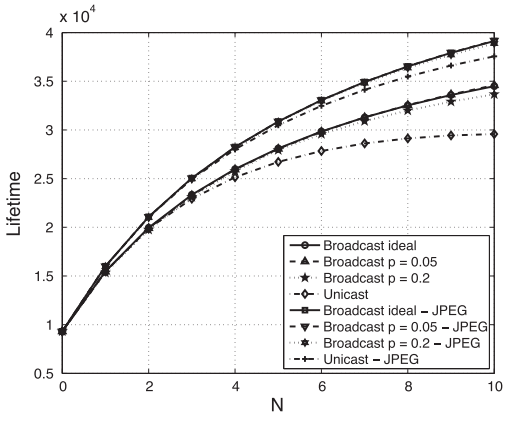
For the considered scenario we have seen that the broadcast offloading scheme performs poorly under heterogeneous link transmission rates. As the broadcast transmission rate is determined by the weakest link, we expect that the performance of the broadcast offloading scheme depends on the difference between the link qualities of the cooperating nodes, and reaches and then outperforms the unicast offloading scheme as this difference decreases.

6.2. Experimental validation

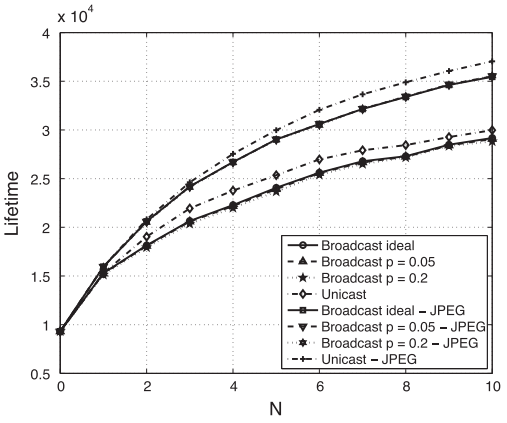
To validate the proposed model, we implemented a visual sensor network composed of one camera node and up to 4 cooperator nodes based on the BeagleBone platform, located in an indoor environment. All nodes run Linux Ubuntu operating system and are equipped with Edimax EW-7811Un WiFi dongles. The camera is a Logitech C170 USB camera. The nodes form a 802.11 Ad Hoc network organized in a star topology, similar to the one in Fig. 3.

6.2.1. Unicast offloading

For unicast offloading the nodes used TCP at the transport layer to ensure reliable transmission. Once the connection is established, the cooperating processing node reports its processing speed v_i and the camera node probes the speed of the link to node i to estimate c_i using the *iperf*

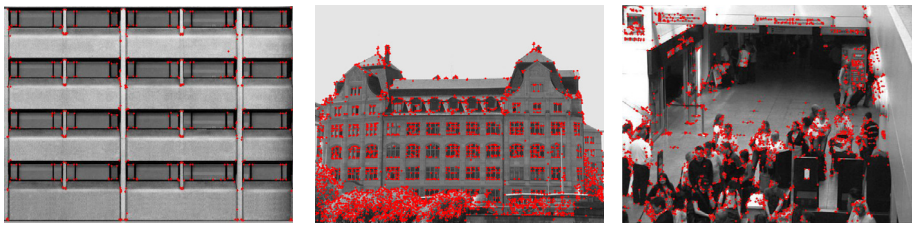


(a) Homoegenous scenario.



(b) Heterogeneous scenario.

Fig. 11. VSN Lifetime vs. number of nodes N.

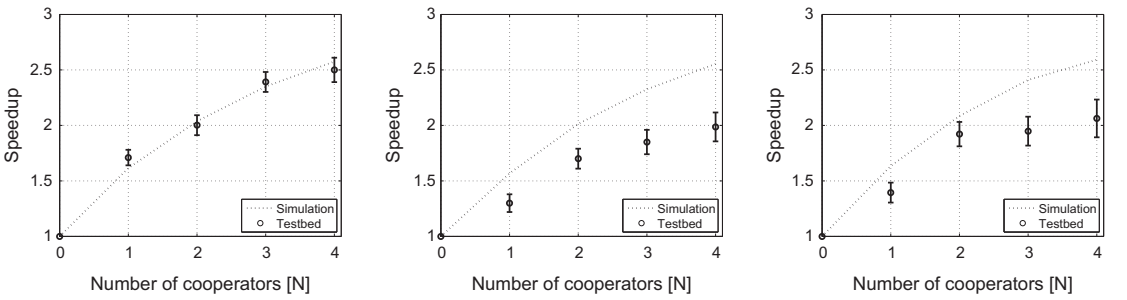


(a) Test image with uniform distribution of key points.

(b) Test image with non-uniform distribution of key-points.

(c) Test image from a real scenario.

Fig. 13. Test images.



(a) Test on the class of images in Figure 13(a).

(b) Test on the class of images in Figure 13(b).

(c) Test on the class of images in Figure 13(c).

Fig. 14. Speedups obtained with the simulation and the testbed for the unicast case.

tool over a window of 10 s. After obtaining the pairs $\langle c_i, \nu_i \rangle$ for the processing nodes, the camera node (i) sorts the processing nodes in descending order of c_i, (ii) computes the optimal slice sizes α_i as described in (5) and (iii) transmits the pixel data to the processing nodes. Upon receiving the data, the processing nodes run the BRISK algorithm on the received image slice and transmit a timestamp corresponding to the completion time back to the camera node. The camera node collects the timestamps from all the nodes and marks the largest one as the completion time. Network Time Protocol (NTP) is used to synchronize the clocks. For each experiment the c_i, ν_i values are recorded, so that the experimental results can be compared with the ones predicted by the analytic framework.

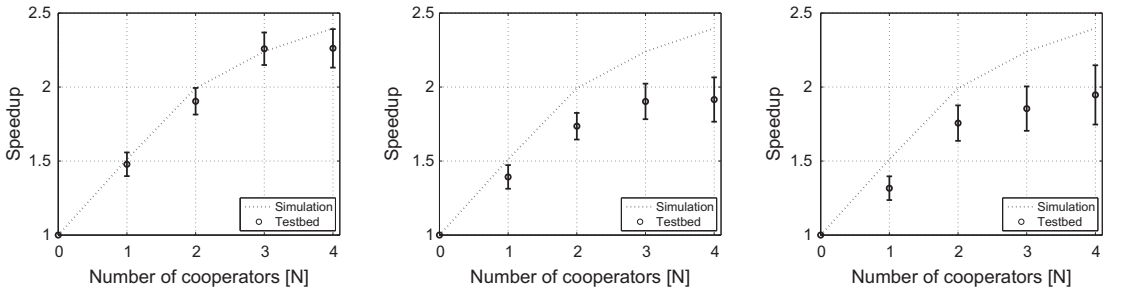
To evaluate the effect of the distribution of the keypoints, we tested our system on three different classes of input images, each class consisting of 10 images. The first class contains images characterized by a uniform distribution of keypoints, as illustrated in Fig. 13(a). The second class is composed of 10 images from the ZuBuD[4] dataset, which contains images of buildings of the city of Zurich (Fig. 13(b)). In this case, the particular visual symmetry of buildings allow to obtain an almost uniform distribution of keypoints, although less balanced. The third class of images, illustrated in Fig. 13(c), is composed of a set of video frames from the PETS 2007[5] surveillance dataset, all characterized by a non-uniform spatial distribution of the keypoints. For each tested image, we repeated the experiment ten times, averaging the results.

Fig. 14 shows the speedup obtained in the testbed as a function of the number of processing nodes. As one can see in Fig. 14(a), in case of a test image characterized with a uniform spatial distribution of keypoints, the speedup obtained with the testbed matches the one predicted by the analytical framework, confirming the validity of the model. Under non-uniform keypoint distribution (Fig. 14(b) and (c)), the model overestimates the speedup achievable with the testbed, due to the relation between the assigned load and the computational time, which in these cases is not linear anymore. As expected, the gap increases with the number of processing nodes. Nonetheless, even for these images, offloading allows to obtain a speedup of 2 when using 4 cooperators, thus decreasing the completion time by 50%. If there is correlation between the keypoint distribution in subsequently processed images, a histogram-based predictive approach can be used to minimize the completion times with good accuracy [33].

6.2.2. Broadcast offloading

To compute the optimal load distribution under broadcast offloading, the camera node needs to estimate (i) the broadcast capacity c_{bc}, (ii) the broadcast error probability p_i and (iii) the pairs $\langle c_i, \nu_i \rangle$ for each node. For what regards c_{bc}, we have observed that the Edimax EW-7811Un WiFi

[4] http://www.vision.ee.ethz.ch/showroom/zubud/index.en.html.
[5] http://www.cvg.reading.ac.uk/PETS2007/data.html.



(a) Test on the class of images in Figure 13(a). (b) Test on the class of images in Figure 13(b). (c) Test on the class of images in Figure 13(c).

Fig. 15. Speedups obtained with the simulation and the testbed for the broadcast case.

dongle used in the testbed works at a fixed and unchangeable broadcast rate of 11 Mb/s, thus c_{bc} has been set to such value. The following process has been used to estimate p_i: first the camera node sends a stream of 1000 packets of fixed size using UDP broadcast communication. Each of the cooperating nodes then reports the number of lost packets, so that p_i can be empirically measured. Finally, the pairs $\langle c_i, v_i \rangle$ are obtained as in the unicast case, i.e., using *iperf* in TCP mode over a window of 10 s.

Fig. 15 shows the speedup achieved in the testbed for broadcast offloading. Again, for an image containing uniformly distributed keypoints, the experimental results match the analytical results, while under non-uniform keypoint distribution the gap is similar to the one in the unicast case.

7. Conclusions and future work

We have shown how cooperation among sensor nodes can be leveraged to minimize the completion time of state-of-the-art algorithms for extracting local features from images, which form the basis for enabling visual analysis tasks in sight-enabled wireless sensor networks. We modeled a scenario in which the camera sensor node offloads part of the visual processing task to a subset of neighboring sensor nodes. We provided closed-form expressions for the optimal offloading solution (ordered sequence of cooperating nodes, load size to be assigned to each cooperation node) and for the overall task processing time by extending results of divisible load theory. We used the performance evaluation framework to evaluate the energy/speedup trade-off involved in the offloading process under different communication strategies (unicast/broadcast). The proposed offloading solutions have been implemented on a real visual sensor network testbed, and were used to validate the performance model and to confirm that considerable speedups may be achievable. Future research avenues may include (i) the extension of DLT theory to cover the transmission of the computed features to the controller and to consider image content dependent processing speed; (ii) the extension to the case where the image slices to be distributed are compressed before transmission; (iii) the joint optimization of the processing time and the energy consumption of the offloading process, also aiming at balancing the energy consumption at the processing nodes; and (iv) the extension to the case where multiple camera nodes need to offload computation to a (partially) common set of cooperating nodes.

Acknowledgments

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 No.: 296676.

References

[1] I.F. Akyildiz, T. Melodia, K.R. Chowdhury, A survey on wireless multimedia sensor networks, Comput. Netw. 51 (4) (2007) 921–960.
[2] S. Soro, W.R. Heinzelman, A Survey of Visual Sensor Networks, Adv. in MM 2009.
[3] A. Redondi, L. Baroffio, M. Cesana, M. Tagliasacchi, Compress-then-analyze vs. analyze-then-compress: two paradigms for image analysis in visual sensor networks, in: 2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP), 2013, pp. 278–282. http://dx.doi.org/10.1109/MMSP.2013.6659301.
[4] S. Leutenegger, M. Chli, R. Siegwart, Brisk: binary robust invariant scalable keypoints, in: ICCV, 2011, pp. 2548–2555.
[5] A. Redondi, L. Baroffio, A. Canclini, C.M., T.M., A visual sensor network for object recognition: testbed realization, in: IEEE/EURASIP Digital Signal Processing Conference. http://dx.doi.org/10.1002/dac.2378, <http://dx.doi.org/10.1002/dac.2378>.
[6] G. Coley, Beaglebone Rev a6 System Reference Manual, 2012.
[7] P. Korshunov, W.T. Ooi, Reducing frame rate for object tracking, in: S. Boll, Q. Tian, L.Z. 0001, Z. Zhang, Y.-P.P. Chen (Eds.), MMM, Lect. Notes Comput. Sci., vol. 5916, Springer, 2010, pp. 454–464.
[8] P. Korshunov, W.T. Ooi, Critical video quality for distributed automated video surveillance, in: H. Zhang, T.-S. Chua, R. Steinmetz, M.S. Kankanhalli, L. Wilcox (Eds.), ACM Multimedia, ACM, 2005, pp. 151–160.
[9] V. Bharadwaj, T.G. Robertazzi, D. Ghose, Scheduling Divisible Loads in Parallel and Distributed Systems, IEEE Computer Society Press, Los Alamitos, CA, USA, 1996.
[10] L.-Y. Duan, X. Liu, J. Chen, T. Huang, W. Gao, Optimizing jpeg quantization table for low bit rate mobile visual search, in: IEEE Visual Communications and Image Processing (VCIP), 2012.
[11] J. Chao, C. Hu, E. Steinbach, On the design of a novel jpeg quantization table for improved feature detection performance, in: 2011 18th IEEE International Conference on Image Processing (ICIP), 2013.
[12] E. Rosten, R. Porter, T. Drummond, Faster and better: a machine learning approach to corner detection, IEEE Trans. Pattern Anal. Mach. Intell. 32 (1) (2010) 105–119.
[13] H. Jegou, M. Douze, C. Schmid, Product quantization for nearest neighbor search, IEEE Trans. Pattern Anal. Mach. Intell. 33 (1) (2011) 117–128.

[14] A. Redondi, L. Baroffio, J. Ascenso, M. Cesana, M. Tagliasacchi, Rate-accuracy optimization of binary descriptors, in: IEEE International Conference on Image Processing, 2013.
[15] V.R. Chandrasekhar, S.S. Tsai, G. Takacs, D.M. Chen, N.-M. Cheung, Y. Reznik, R. Vedantham, R. Grzeszczuk, B. Girod, Low latency image retrieval with progressive transmission of chog descriptors, in: Proceedings of the 2010 ACM Multimedia Workshop on Mobile Cloud Media Computing, MCMC '10, ACM, New York, NY, USA, 2010, pp. 41–46. http://dx.doi.org/10.1145/1877953.1877966, <http://doi.acm.org/10.1145/1877953.1877966>.
[16] A. Redondi, M. Cesana, M. Tagliasacchi, Rate-accuracy optimization in visual wireless sensor networks, in: 2012 19th IEEE International Conference on Image Processing (ICIP), 2012, pp. 1105–1108. http://dx.doi.org/10.1109/ICIP.2012.6467057.
[17] M. Moges, T. Robertazzi, Wireless sensor networks: scheduling for measurement and data reporting, IEEE Trans. Aerosp. Electron. Syst. 42 (1) (2006) 327–340, http://dx.doi.org/10.1109/TAES.2006.1603426.
[18] X. Li, X. Liu, H. Kang, Sensing workload scheduling in sensor networks using divisible load theory, in: Global Telecommunications Conference, 2007, GLOBECOM '07, IEEE, 2007, pp. 785–789. http://dx.doi.org/10.1109/GLOCOM.2007.152.
[19] S. Bataineh, T. Hsiung, T.G. Robertazzi, Closed form solutions for bus and tree networks of processors load sharing a divisible job, in: International Conference on Parallel Processing, 1993, ICPP 1993, vol. 1, 1993, pp. 290–293. http://dx.doi.org/10.1109/ICPP.1993.54.
[20] V. Bharadwaj, D. Ghose, V. Mani, Optimal sequencing and arrangement in distributed single-level tree networks with communication delays, IEEE Trans. Parallel Distrib. Syst. 5 (9) (1994) 968–976, http://dx.doi.org/10.1109/71.308534.
[21] L. Anand, D. Ghose, V. Mani, Analysis of the drop-out rule in probabilistic load sharing in distributed computing systems, Int. J. Syst. Sci. 28 (5) (1997) 457–465.
[22] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vision 60 (2) (2004) 91–110, http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94. <http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94>.
[23] H. Bay, A. Ess, T. Tuytelaars, L.J.V. Gool, Speeded-up robust features (surf), Comput. Vis. Image Underst. 110 (3) (2008) 346–359.
[24] E. Mair, G.D. Hager, D. Burschka, M. Suppa, G. Hirzinger, Adaptive and generic corner detection based on the accelerated segment test, in: ECCV (2), 2010, pp. 183–196.
[25] M. Calonder, V. Lepetit, M. Özuysal, T. Trzcinski, C. Strecha, P. Fua, Brief: Computing a local binary descriptor very fast, IEEE Trans. Pattern Anal. Mach. Intell. 34 (7) (2012) 1281–1298.
[26] J. Heinly, E. Dunn, J.-M. Frahm, Comparative evaluation of binary features, in: Computer Vision–ECCV 2012, Springer, 2012, pp. 759–773.
[27] A. Canclini, M. Cesana, A. Redondi, M. Tagliasacchi, J. Ascenso, R. Cilla, Evaluation of low-complexity visual feature detectors and descriptors, in: IEEE/EURASIP Digital Signal Processing Conference.
[28] J. Chao, E. Steinbach, Preserving sift features in jpeg-encoded images, in: 2011 18th IEEE International Conference on Image Processing (ICIP), 2011, pp. 301–304. http://dx.doi.org/10.1109/ICIP.2011.6116299.
[29] J. Chao, A. Al-Nuaimi, G. Schroth, E. Steinbach, Performance comparison of various feature detector-descriptor combinations for content-based image retrieval with JPEG-encoded query images, in: IEEE International Workshop on Multimedia Signal Processing (MMSP), Pula, Sardinia, Italy, 2013.
[30] M. Khan, G. Dan, V. Fodor, Characterization of surf interest point distribution for visual processing in sensor networks, in: 2013 18th International Conference on Digital Signal Processing (DSP), 2013, pp. 1–7. http://dx.doi.org/10.1109/ICDSP.2013.6622701.
[31] K. Mikolajczyk, C. Schmid, Scale & affine invariant interest point detectors, Int. J. Comput. Vision 60 (1) (2004) 63–86, http://dx.doi.org/10.1023/B:VISI.0000027790.02288.f2.
[32] B. Veeravalli, X. Li, C.-C. Ko, On the influence of start-up costs in scheduling divisible loads on bus networks, IEEE Trans. Parallel Distrib. Syst. 11 (12) (2000) 1288–1305, http://dx.doi.org/10.1109/71.895794.
[33] E. Eriksson, G. Dán, V. Fodor, Real-time distributed visual feature extraction from video in sensor networks, in: Proc. of IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), 2014.

Alessandro Redondi received his Master degree in Computer Engineering in July 2009 and his Ph.D. in Information Engineering in February 2014, both from Politecnico di Milano. From September 2012 to April 2013 he worked as a visiting researcher at the University College of London. Currently he is an Assistant Researcher at the Department of Electronics and Information of the Politecnico di Milano. His research activities are focused on algorithms and protocols for Real Time Localization Systems and Wireless Multimedia Sensor Networks.

Matteo Cesana received his MS degree in Telecommunications Engineering and his Ph.D. degree in Information Engineering from the Politecnico di Milano in July 2000 and in September 2004, respectively. From September 2002 to March 2003 he has been working as a visiting researcher at the Computer Science Department of the University of California in Los Angeles (UCLA). He is now an Assistant Professor of the Electronics and Information Department of the Politecnico di Milano. His research activities are in the field of performance evaluation of cellular systems, ad hoc networks protocol design and evaluation and wireless networks optimization. He is an Associate Editor of Ad Hoc Networks Journal (Elsevier).

Marco Tagliasacchi is currently Assistant Professor with the Dipartimento di Elettronica e Informazione–Politecnico di Milano, Italy. He received the Laurea degree (2002, summa cum Laude) in computer engineering and the Ph.D. degree in electrical engineering and computer science (2006), both from Politecnico di Milano.

He was a visiting academic at the Imperial College London (2012) and visiting scholar at the University of California, Berkeley (2004). His research interests include multimedia communications (coding, quality assessment), multimedia forensics, and information retrieval. He coauthored more than 100 papers in international journals and conferences, including award winning papers at MMSP 2012, ICIP 2011, MMSP 2009, and QoMex 2009. He has been actively involved in several EU-funded research projects. He is currently co-coordinating two ICT-FP7 FET-Open projects (REWIND and GreenEyes).

He is an elected member of the IEEE MMSP Technical Committee for the term 2009–2012. He serves as Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGIES (2011 best AE award) and APSIPA Transactions on Signal and Information Processing. He served in the Organizing Committee of the ACM/IEEE Inter- national Conference on Distributed Smart Cameras (ICDSC) 2009 and Digital Audio Effects (DAFx 2009). He will be General co-Chair of IEEE Workshop on Multimedia Signal Processing (MMSP 2013, Pula, Italy) and Technical Program Coordinator of IEEE International Conference on Multimedia & Expo (ICME 2015, Turin, Italy).

Ilario Filippini received the B.S. and M.S. degrees in telecommunication engineering and Ph.D. degree in information engineering from the Politecnico di Milano, Milan, Italy, in 2003, 2005, and 2009, respectively.
From February to August 2008, he worked as a Visiting Researcher with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus. He is currently an Assistant Professor with the Electronics and Information Department, Politecnico di Milano. His research activities include networking and optimization issues, in particular wireless multihop network planning, optimization, and protocol design.

György Dán is an associate professor at KTH Royal Institute of Technology, Stockholm, Sweden. He received the M.Sc. degree in computer engineering from the Budapest University of Technology and Economics, Hungary in 1999, the M.Sc. degree in business administration from the Corvinus University of Budapest, Hungary in 2003, and the Ph.D. in Telecommunications from KTH in 2006.
He worked as a consultant in the field of access networks, streaming media and videoconferencing 1999–2001. He was a visiting researcher at the Swedish Institute of Computer Science in 2008, and a Fulbright research scholar at the Information Trust Institute at University of Illinois Urbana-Champaign in 2012–2013. His research interests include the design and analysis of content management and computing systems, game theoretical models of networked systems, and cyber-physical system security in power systems.

Viktoria Fodor received her M.Sc. and Ph.D. degrees in computer engineering from the Budapest University of Technology and Economics in 1992 and 1999, respectively. In 1999 she joined KTH Royal Institute of Technology, where she now acts as associate professor in the Laboratory for Communication Networks. Her current research interests include network performance evaluation, protocol design, sensor and multimedia networking.