Networked Computer Vision: the importance of a holistic simulator

Juan C. SanMiguel and Andrea Cavallaro

WiSE-Mnet++ is a holistic simulator that abstracts the key functions of smart-camera networks and models the main operations to account for hardware capabilities, the complexities of visual data, and their associated high-data-rate communication.

***Index Terms*—Visual sensor networks, smart cameras, simulator, distributed, resource consumption.**

I. Introduction

Smart-camera networks (SCNs) enable a range of services for vehicular ad hoc networks, smart cities, home automation, wide-area surveillance, and search-and-rescue operations. With built-in processing and communication capabilities, these camera networks generate large volumes of data, share high-data-rate messages, and generally operate with limited resources.

The success of SCNs depends on the availability of simulators that facilitate fast algorithmic prototyping and validate performance objectives before deployment. Simulation tools can help predict performance and provide feedback on the models to be employed in real-world systems. Such tools must account for the myriad of operational conditions and heterogeneity of devices that compose a SCN. Although early work on camera networks assumed infinite bandwidth or cost-free data exchange,[1] real-world SCNs must consider the constraints imposed by resource-limited platforms. For example, battery-powered cameras on self-driving vehicles must wirelessly communicate with main-powered static cameras to track pedestrians without exhausting their energy and the available bandwidth.

Because cameras capture, process, and transmit much larger volumes of data than traditional sensor networks, they present unique design and operational challenges, which existing simulators lack the necessary functionalities to evaluate. (See the "Camera-Network Simulators" sidebar for more details.) Designing SCN simulators requires interdisciplinary expertise covering algorithms, hardware, and networking in order to model the camera hardware, identify appropriate resources, and emulate communication protocols and channels[2].

To address these challenges and simulate a range of application scenarios, we developed the WiSE-Mnet++ simulator. Our simulator models the key operations in smart cameras (sensing, processing, communication, and decision making), offers power-consumption models for smart-camera hardware, and simulates realistic multicamera networks with both real-world and synthetic datasets.

Juan C. SanMiguel is with Universidad Autónoma de Madrid (Spain), email: juancarlos.sanmiguel@uam.es. Andrea Cavallaro is with Queen Mary University of London (UK), e-mail: a.cavallaro@qmul.ac.uk

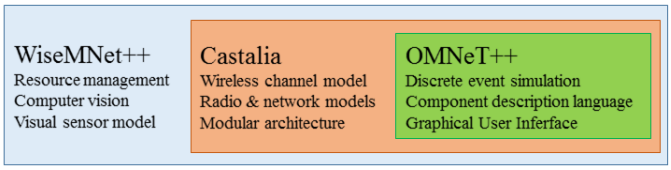


Figure 1: The relation between the WiSE-Mnet++ camera-network simulator, Castalia (http://castalia.forge.nicta.com.au/) and OMNeT++ (http://omnetpp.org/).

The WiSE-Mnet++ open source simulator is available to the research community at www.eecs.qmul.ac.uk/~andrea/wise-mnet.html, along with supplementary material describing how to incorporate new simulation features and SCN algorithms. The WiSE-Mnet++ simulator facilitates smart-camera research by enabling users to activate or deactivate each simulated feature. They can also easily compare solutions for specific research problems, such as the impact of real communication channels or limited computing capabilities on performance.

In this article, we discuss the WiSEMnet++ simulator's main features and provide two examples that show its effectiveness in profiling performance and energy consumption for networked computer-vision applications.

II. Camera Node

WiSE-Mnet++ provides generic, yet descriptive modeling of the camera operations for sensing, processing, and communication. As Figure 1 illustrates, WiSE-Mnet++ extends the WiSE-Mnet simulator[3]. and is based on the OMNeT++ (http://omnetpp.org/) and Castalia SN (http://castalia.forge.nicta.com.au/) simulators.

A smart camera consists of layers that cover specific functionalities (see Figure 2a). A layer's functionality can be easily extended following an object-oriented scheme. The hardware associated with each layer is also simulated to determine the camera operational capabilities (such as processing frequency) and resources (such as battery power).[4] A message-passing structure enables interlayer communication.

A. Sensing

The WiseBaseSensor layer provides input data by measuring the physical phenomena observed by the camera network. To introduce new sensor functionalities, we can extend this layer with sublayers, such as the WiseCameraManager, to control the sensing and capturing parameters, including focal length.

CAMERA-NETWORK SIMULATORS

Early simulators of camera networks focused primarily on the use of video datasets for multi-camera surveillance and sport games (http://datasets.visionbib.com/). More comprehensive simulators were later proposed to account for communication and coordination with smart cameras. Table I summarizes these simulators that can be classified as local or global.

Local simulators test a particular aspect of cameras. For example, the Object Video Virtual Tool (OVVT) [a] and the Software Laboratory for Camera Network Research (SLCNR) [b] use virtual worlds to emulate the sensing of real-life scenarios. The Visual Sensor Network simulator (VSNSim) [c] also supports coordination and control, but lacks models for camera resources and communication channels thus making it difficult to implement realistic coordination approaches. Moreover, extending the functionalities of these simulators is not straightforward as they are provided as bundled packages. Finally, the CamSim simulator [d] defines protocols for communication between cameras, but without realistic communication models and without real-world video data as input.

Global simulators focus on realistic camera networking by extending OMNeT++, a popular discrete-event simulator for Wireless Sensor Networks. The Wireless Video Sensor Network (WVSN) simulator [e] determines the visual coverage of cameras over static 2D images, but without using video streams or visual analytics. The Mobile MultiMedia Wireless Sensor Network (M3WSN) [f] simulator addresses multimedia transmission without enabling collaborative processing. Although these simulators are extensible and can use communication protocols, they are mainly focused on 2D measurements, without support for video data, visual tools or resource-consumption models for smart-camera platforms.

WiSE-Mnet++, our smart-camera network simulator, takes advantage of discrete-event simulation to address the above-mentioned shortcomings.

Ref	Name	Type	Calibration	Camera	Sensing		Processing		Communication		Coordination		Resources		Extensible
				Mobility	Synthetic	Real	Scalable	Visual	Ideal	Realistic	Topology	Modes	Consumption	Allocation	
[a]	OVVT	CS	✓	✓	V			✓				SY			
[b]	SLCNR	CS	✓		V			✓				SY			✓
[c]	VSNSim	CS			V			✓	✓			SY			
[d]	CamSim	DS			MP				✓		CG, VG	SY			✓
[e]	WVSN	DS		✓	MP	I	✓			✓	CG	SY	C	S	✓
[f]	M3WSN	DS		✓	MP	I				✓	CG	SY	C	S	✓
Ours	**WiSE-Mnet++**	**DS**	✓	✓	**V,MP**	**I,R,L**	✓	✓	✓	✓	CG,VG	AS, SY	P	D	✓

Table I: Simulators for smart-camera networks and their main features. Empty cells represent features not offered by the corresponding simulator. KEY -- CS: Continuous simulation (real time). DS: Discrete Simulation. MP: Moving Points. V: Virtual video. R: Recorded video. L: Live video. CG: Communication Graph. VG: Vision Graph. AS: Asynchronous. SY: Synchronous. C: Constant. P: Parametric. S: Static. D: Dynamic.

REFERENCES

[a] G. Taylor et al., "OVVV: Using Virtual Worlds to Design and Evaluate Surveillance Systems", IEEE Conf. on Computer Vision and Pattern Recognition, pp. 1-8, Jun. 2007. Available: http://development.objectvideo.com/

[b] W. Starzyk and F. Qureshi, "Software Laboratory for Camera Networks Research", IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 3, no. 2, pp. 284-293, Feb. 2013. Available: https://github.com/vclab/virtual-vision-simulator

[c] M. Gruber et al., "Demo: The extended vsnsim for hybrid camera systems", in Int. Conf. on Distributed Smart Cameras, pp. 203-204, Sept. 2015

[d] L. Esterle et al., "CamSim: A Distributed Smart Camera Network Simulator", in IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems Workshops, pp. 19-20, Sept. 2013. Available: https://github.com/EPiCS/CamSim

[e] C. Pham and A. Makhoul, "Performance study of multiple cover-set strategies for mission-critical video surveillance", IEEE Int. Conf. on Wireless and Mobile Computing, Networking and Comms., pp. 208-216, Oct. 2010. Available: http://cpham.perso.univ-pau.fr/WSN-MODEL/wvsn.html

[f] D. Rosario et al., "An OMNeT ++ Framework to Evaluate Video Transmission in Mobile Wireless Multimedia Sensor Networks", ICST Conf. on Simulation Tools and Techniques, pp. 277-284, Mar. 2013. Available: http://home.inf.unibe.ch/~zhao/M3WSN/

The WiseBasePhysicalProcess layer defines the observable phenomena (see Figure 3). The WiseVideoFile and WiseVirtualCam extensions let us use video data from real-world datasets and from virtual 3D worlds such as Unity (unity3d.com). Moreover, we can model synthetic objects as simple moving points on a common coordinate system (such as ground plane or zenithal view) via the WiseMovingTarget extensions. In this case, we model the directional sensing of the field of view (FoV) on the ground plane as a 2D polygon defined by the orientation, angle, and depth of the camera view.

Unlike Pan-Tilt-Zoom smart cameras that consider only dynamic FoVs, the WiseBaseMobility enables to spatially move cameras by simulating the physical motion of their location that is typical of vision-based robotic applications.[5]

B. Processing

The processing of video streams is pivotal for decision making and WiSE-Mnet++ defines a hierarchy of modules to coordinate the execution of the camera operations. The WiseBaseApplication layer is the interface with the network and provides basic capabilities to exchange data via the WiseBaseComm layer. The WiseCameraAlgorithm layer extends WiseBaseApplication with functions running at initialization and others called periodically for receiving new data. These functions also define a finite-state-machine that sequentially performs the three main camera operations for each sensed sample (e.g. a video frame). OMNeT++ timers are used to specify response times of the processing capabilities and to control the frequency when collecting data from WiseBaseSensor. Moreover, the sub-layer WiseCameraPeriodicTracker provides a ready-to-use functionality for target tracking. Finally, user applications are implemented by extending WiseCameraAlgorithm or WiseCameraPeriodicTracker with custom video analysis tools or third party libraries such as OpenCV.

C. Communication

Unsynchronized and instantaneous inter-camera communication is enabled by toNodeDirect gates defined inside each WiseNode camera. This direct communication is useful for testing algorithms without considering the network. The communication protocols and channels are implemented in the WiseBaseComm layer, which considers both ideal and realistic

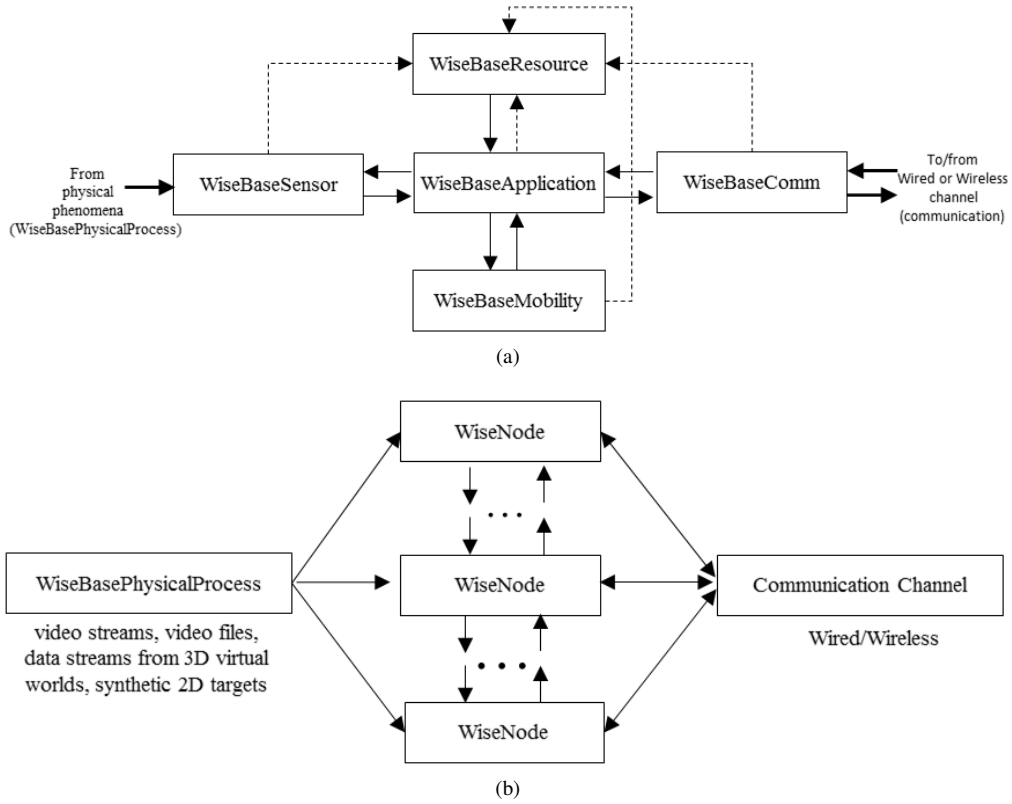


Figure 2: Layered WiSE-Mnet++ simulation for smart-camera networks. (a) A smart-camera node (`WiseNode`). Sensing, processing and communication capabilities are handled by the `WiseBaseSensor`, `WiseBaseApplication` and `WiseBaseComm` layers, respectively. The `WiseBaseMobility` changes the camera location and the `WiseBaseResource` monitors the employed resources. (b) Within an SCN, WiseNode cameras are interconnected via wired-wireless channels or direct (instantaneous) message passing.

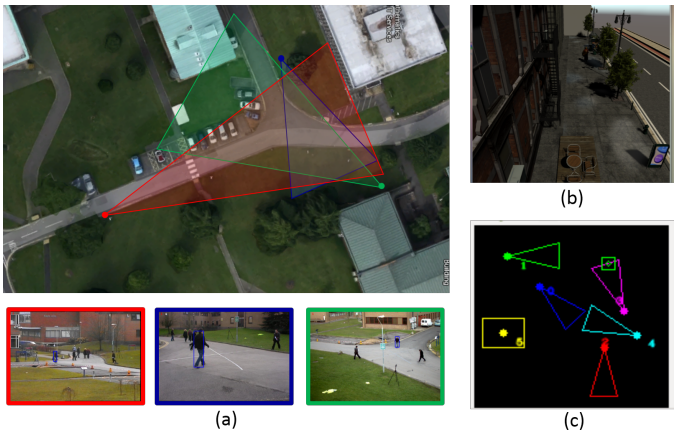


Figure 3: The three sensing options available in WiSEMnet++: (a) real-world video input or pre-recorded sequences (PETS 2009 dataset http://www.cvg.reading.ac.uk/PETS2009/); (b) streams from virtual 3D worlds; and (c) 2D synthetic data.

communication modes for data exchange. Buffer structures are defined to store the received data.

The ideal communication is an idealization of wired communications that helps develop collaborative algorithms while avoiding network- and transceiver-related problems when exchanging data, such as collisions resulting from multiple cameras simultaneously transmitting. The WiseDummyWirelessChannel layer bypasses the communication protocol stack and enables a synchronized connectivity among cameras. The simulator also provides ideal communication conditions with instantaneous data exchanges without any packet losses or interferences.

The Castalia simulator provides the realistic communication. Castalia defines transceiver models (radio), advanced channel models (WirelessChannel), and routing protocols for wireless sensor networks implemented in the VirtualMac layer. Realistic conditions should account for multiple factors such as the transceiver (radio) models, the communication protocol (such as MAC), interference and attenuation of the wireless channel, and the latencies of the camera modules.

D. Resource management

The WiseBaseResource layer models the resources and consumption associated to camera hardware, which is key for resource-aware camera networks.[6] This layer also reports usage statistics to WiseBaseApplication for further reasoning. For example, a camera may re-allocate a task to other cameras to extend its lifetime.

WiSE-Mnet++ provides capability descriptors to model common hardware features, such as frame rate and frame size

for sensing, memory and operating frequency for processing, and available bandwidth and power modes for communication. The WiseBaseResource layer loads these descriptors when initializing the simulation. We can incorporate new hardware features by extending this layer.

To model energy consumption, each camera layer operates with a three-state model.[7] A specific state (active, sleep, or idle) can be selected on demand (such as when the processor is asked to complete a task) or via designer-defined rules (such as by forcing a camera to sleep after a certain period of idleness). We approximate the power of the active state using an N-order polynomial model that accommodates existing nonlinearities between resource usage and consumption. We model the power for the sleep and idle states as constants.

III. Camera network

Networked computer vision involves several cameras communicating with each other via single or multiple hops. WiSE-Mnet++ identifies the intercamera links to enable the control of such networks (see Figure 2b).

A. Network topology

WiSE-Mnet++ describes the network topology based on two types of neighborhood connectivity: vision and communication. The *vision neighborhood* defines cameras that share a portion of their FoV. The *communication neighborhood* determines cameras that can exchange messages with a single-hop communication. This neighborhood information can be manually introduced or automatically discovered.

The WiseCameraAlgorithm layer can automatically compute the vision connectivity using external camera calibration data (that is, the camera location and orientation on a common coordinate system such as the ground plane). The automatic discovery of communication connectivity relies on an iterative send-and-receive protocol performed in the WiseBaseApplication layer. However, researchers can easily add more complex online approaches (such as task exchange patterns[8]) to discover and adapt the knowledge of the network topology during runtime.

B. Collaboration modes

The WiseCameraAlgorithm template supports two operation modes - asynchronous and synchronous - that can be selected in the initialization phase.

Asynchronous duty-cycled camera networks allow faster response times. In this case, cameras are always ready to collaborate, and camera operations are not temporally coordinated. Hence, sensing acquires frames at a desired frame rate, and the communication layer continuously listens to the channel for incoming data. Buffers are used for both sensing and communication as the data sensed or received could be processed with a delay. Processing is triggered when any of the buffers contains data.

In the synchronous mode, cameras iteratively perform sequential sensing, processing, and communication. No buffering is required because each operation starts after the previous one finishes. The execution pipeline's speed is therefore determined by its slowest operation, which potentially limits the responsiveness of the entire SCN during collaboration.

IV. Case studies

We illustrate the advantages of WiseMNet++ using two important SCN applications: person reidentification and distributed tracking. For the smart camera hardware, we used an ARM-A9 processor (0.5-1.5 GHz), a B3 image sensor (10-24 MHz), and a C2420 radio (250 Kbps).[7] The simulations were performed on a PIV-3.1 GHz with 4 Gbytes of RAM.

A. People descriptors (in-node processing)

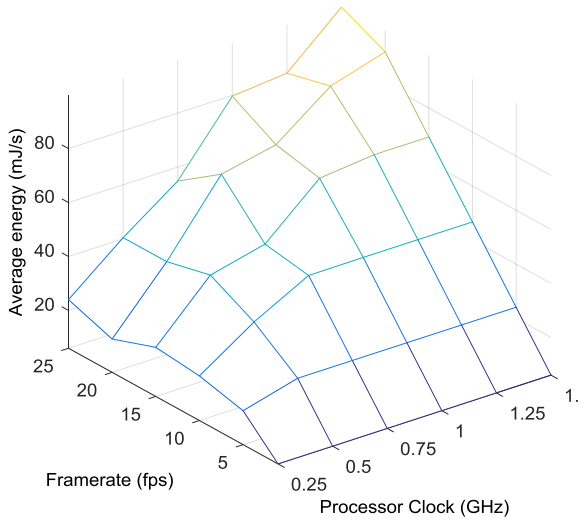
We can begin by profiling the energy consumption of a detect-describe-transmit task for person reidentification.[9] In doing so, we vary the sensing frame rate and the processing clock frequency. Each camera detects people within its FoV and generates visual descriptors of their appearance. For each frame, people are described by a vector including synchronization data (timestamp), the number of detections, normalized RGB histograms (three channels, 16 bins/channel, and 256 levels/bin), and spatial descriptors (center coordinates and the bounding box's width and height). Each detection generates a 6,600-bit packet, which is compressed using Huffman encoding. We customized the WiseCameraApplication sublayer to implement the described functionality, and the camera employs video files using the WiseVideoFile extension.

Figure 4 reports the results for the *AVSS07_AB_eval*sequence (www.avss2007.org). Figure 4a shows the energy consumption of the communication layer as a function of the camera hardware capabilities. High frame rates and high processor speeds lead to an energy consumption that is only one order of magnitude smaller than that of processing. Moreover, Figure 4b and Figure 4c show the energy consumption rate for the processing module's active and idle states. The energy required for processing depends on both the frame rate and clock frequency. When we combine the idle and active states, the consumption ranges from 25 mW (0.25 GHz) to 870 mW (1.5 GHz). As we increase the clock frequency, frames are processed faster and the associated cost increases. The idle state's energy is only relevant when the processor is not loaded (1-5 fps) and operates at high frequencies (0.75-1.5GHz), which is comparable to the active energy. This is interesting because it shows that, despite current assumptions, the idle power must be considered when measuring power consumption.

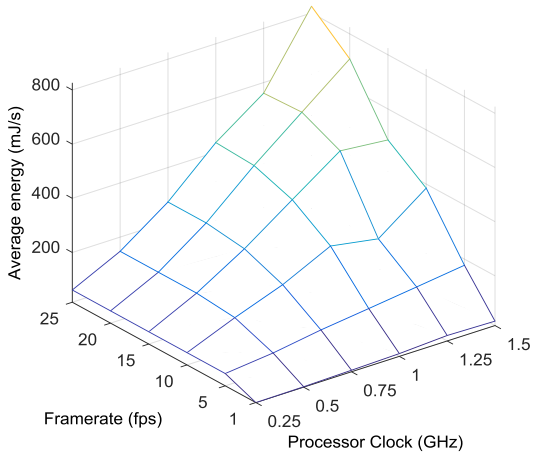
B. Distributed tracking (in-network processing)

For our second case study, consider a distributed fusion task with cameras exchanging data without the coordination of a task leader. Here, we use a wireless camera network with eight cameras that cover a $500m \times 500m$ area. The cameras obtain measurements at 4 Hz (that is, a sampling time of 0.25 s) and have a communication range of 250 m. Each target moves for 40 s.

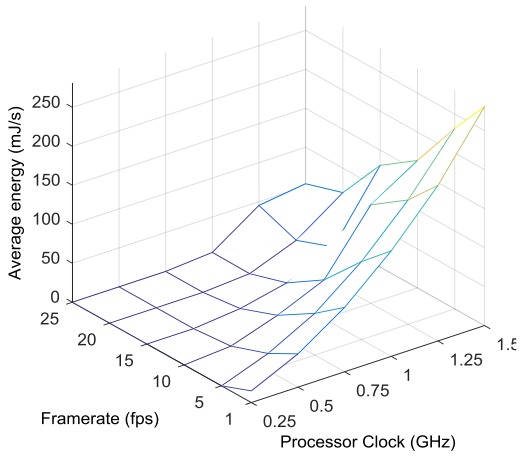
We can apply a consensus-based approach to distributively achieve an average quantity among the network nodes. In



(a) Communication: energy consumption



(b) Processing: active energy consumption



(c) Processing: idle energy consumption

Figure 4: Energy consumption of the *detect-describe-transmit* task for the (a) communication module; (b) processing module (active state) and (c) processing module (idle state). Note that for high processor clocks and low frame rates the consumption of the idle and active states are comparable.

such an iterative scheme, the nodes reach a consensus by sharing the data and then computing the mean of the received quantities. We perform consensus-based single- and multiple-target tracking (MTT) and measure the accuracy, energy consumption, and delay associated with processing in ideal and realistic network conditions over 200 independent runs. We adapt the WiseCameraPeriodicTracker sublayer to perform consensus and use the WiseMovingTarget extension to sense moving targets within the FoV of cameras.

For single-target tracking, we compare two consensus-based approaches: the Kalman-consensus filter (KCF)[10] and the information-consensus filter (ICF).[11] Each camera runs a KCF or ICF, and the output is broadcast to all neighboring cameras, which apply consensus to estimate the target state (such as its position on the ground plane).

Under ideal network conditions, as expected, the tracking error decreases when increasing number of iterations as the estimation error of each camera is diffused over the other cameras (see Figure 5a). KCF performs a *blind average* of the target state and therefore accumulates errors of cameras far away from the target. ICF outperforms KCF by sharing prior information about the absence of measurements when the targets are outside the cameras' FoV.

Under realistic conditions, the tracking error for ICF and KCF does not decrease when the number of iterations increases (see Figure 5b). This is due to the accumulated delay for the iterations, because packet transmission and reception do not occur instantly, even for the small packets of ICF (36 bytes) and KCF (18 bytes).

Under ideal conditions, ICF's improvement comes at an extra cost in terms of processing and communication. ICF requires more than twice the energy of KCF for all iterations (see Figure 5c). Note that although research on smart cameras has traditionally considered communication costs negligible

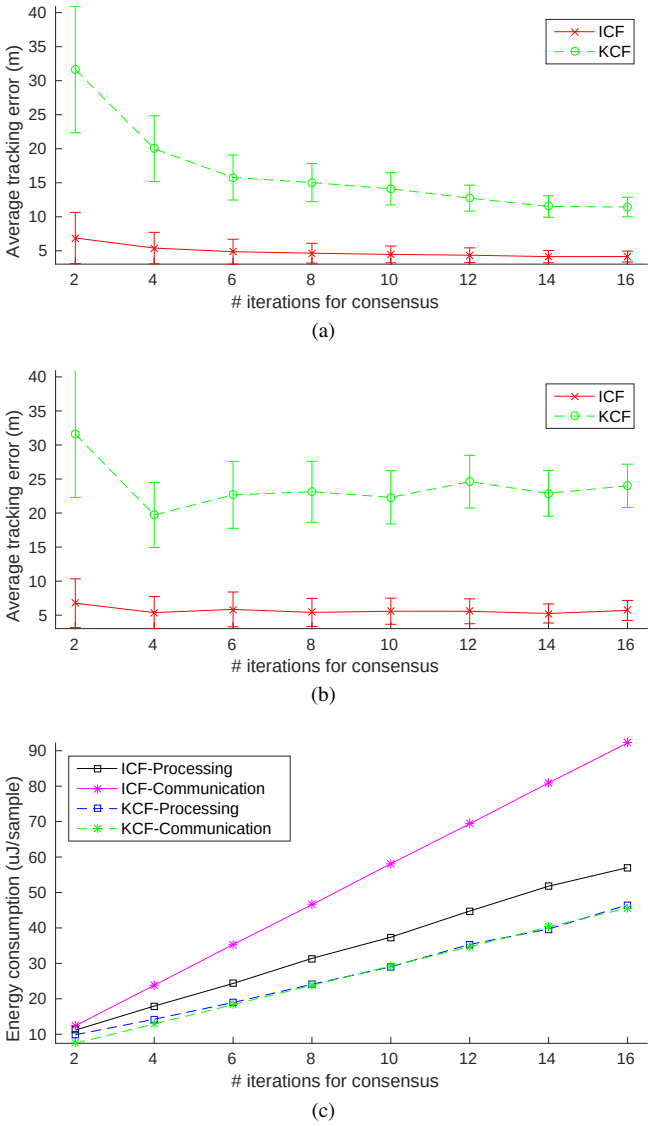


Figure 5: Consensus-based distributed tracking in a single-target tracking network using a Kalman-consensus filter (KCF) and an information-consensus filter (ICF): (a) ideal and (b) realistic wireless communication channels. The error decrease visible under ideal conditions is not maintained under realistic networks because of processing and transmission delays. (c) The average energy consumption for all cameras reveals that communication costs are not always negligible.

compared with that of processing, Figure 5c shows equal costs for KCF, whereas for ICF the cost of communication is greater than that of processing.

For multi-target tracking (MTT), we analyze the MTIC filter[12] , which extends ICF to multiple targets. Network parameters, such as the MAC synchronization window, are configured to the setting that provides the fastest communication without error, which depends on the maximum number of targets (12) for the test conditions. With WiSE-Mnet++ we can explore two key factors affecting the MTT performance: measurements with clutter and network delay.

Figure 6a shows the tracking error for MTIC for various

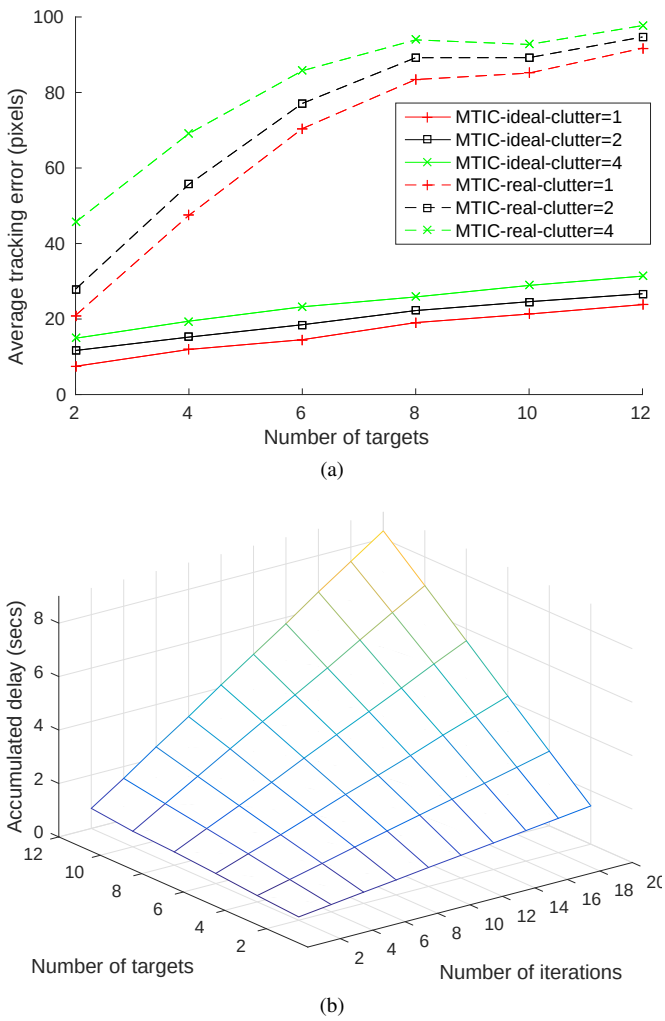


Figure 6: Consensus-based distributed tracking in a multiple-target tracking network. Our analysis of the MTIC filter reveals that (a) the tracking error depends on the delay in real networks and (b) for two targets and six iterations, the delay in processing one sample exceeds the sampling rate (0.25 s).

clutter levels in ideal and realistic communication conditions. As the number of targets grows, it takes longer to exchange target states, thus producing a delay that increases the tracking error (Figure 6b). After the sixth iteration for two targets, the accumulated delay is greater than 0.25 s (the sampling frequency) and therefore cameras miss target measurements. This latency in processing the samples increases the final error of the estimation, regardless of the number of consensus iterations. Considering Figures 6a and 6b, MTIC is more affected by network delays than by clutter, a comparison that is not usually performed when reporting tracking results.[12]

V. Conclusions

WiSE-Mnet++ offers tools that help identify shortcomings and bottlenecks when designing or adopting algorithms for real SCNs so they can be identified before deployment. It is extensible and flexible, and readily allows users to incorporate new features at the algorithm, network, and hardware levels.

Future work will focus on complementing WiSE-Mnet++ with simulation environments that extend the range of available testing scenarios for distributed computer vision algorithms.

ACKNOWLEDGMENTS

Juan C. SanMiguel acknowledges the support of the Spanish Government (HA-Video TEC2014-5317-R). Andrea Cavallaro acknowledges the support of the Artemis JU and the UK Technology Strategy Board (Innovate UK) through the COPCAMS Project under grant 332913.

REFERENCES

1. H. Aghajan and A. Cavallaro, Multi-camera Networks: Principles and Applications, Academic Press, 2009.
2. M. Reisslein, B. Rinner, and A. Roy-Chowdhury, "Guest Editors'Introduction: Smart Camera Networks", Computer, vol. 47, no. 5, 2014, pp. 23-25.
3. C. Nastasi and A. Cavallaro, "WiSE-MNet: An Experimental Environment for Wireless Multimedia Sensor Networks," Proc. Sensor Signal Processing for Defence (SSPD 11), 2011, pp. 34-34.
4. J. Schlessman and M. Wolf, "Tailoring Design for Embedded Computer Vision Applications," Computer, vol. 48, no. 5, 2015, pp. 58-62.
5. B. Bhanu et al., "Guest Editors' Introduction: Distributed Smart Sensing for Mobile Vision," IEEE Sensors J., vol. 15, no. 5, 2015, pp. 2631-2631.
6. C. Piciarelli et al., "Dynamic Reconfiguration in Camera Networks: A Short Survey," IEEE Trans. Circuits and Systems for Video Technology, vol. 5, no. 26, 2015, pp. 965-977.
7. J.C. SanMiguel and A. Cavallaro, "Energy Consumption Models for Smart-Camera Networks," preprint, IEEE Trans. Circuits and Systems for Video Technology, 2016; doi:10.1109/TCSVT.2016.2593598.
8. P. Lewis et al., "Static, Dynamic, and Adaptive Heterogeneity in Distributed Smart Camera Networks," ACM Trans. Autonomous and Adaptive Systems, vol. 10, no. 2, 2015, article no. 8.
9. R. Mazzon, S. Tahir, and A. Cavallaro, "Person Re-identification in Crowd," Pattern Recognition Letters, vol. 33, no. 14, 2012, pp. 1828-1837.
10. R. Olfati-Saber, J.A. Fax, and R.M. Murray, "Consensus and Cooperation in Networked Multi-agent Systems" Proc. IEEE, vol. 95, no. 1, 2007, pp. 215-233.
11. A. Kamal, J. Farrell, and A. Roy-Chowdhury, "Information Weighted Consensus Filters and Their Application in Distributed Camera Networks," IEEE Trans. Automatic Control, vol. 58, no. 12, 2013, pp. 3112-3125.
12. A. Kamal et al., "Distributed Multi-target Tracking and Data Association in Vision Networks," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 7, no. 38, 2016, pp. 1397-1410.

Juan C. SanMiguel is an assistant professor at the University Autonóma of Madrid. His research interests include multicamera activity understanding. SanMiguel received a PhD in electrical engineering from the University Autonoma of Madrid. He is a member of IEEE. Contact him at juancarlos.sanmiguel@uam.es

Andrea Cavallaro is a professor of multimedia signal processing and director of the Centre for Intelligent Sensing at Queen Mary University of London. His research interests include smart camera networks and behavior recognition. Cavallaro received a PhD in electrical engineering from the Swiss Federal Institute of Technology (EPFL), Lausanne. He is a member of IEEE. Contact him at a.cavallaro@qmul.ac.uk.