2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Skinware: A Real-Time Middleware for Acquisition of Tactile Data from Large Scale Robotic Skins

Shahbaz Youssefi, Simone Denei, Fulvio Mastrogiovanni[1], Giorgio Cannata

***Abstract*— Within the past decade, extensive research has been done on large-scale tactile sensing, as a result of which, a large variety of robot skins have been developed. These robot skins are different in various aspects: the sensing modality, interconnectivity of the sensors, modularity, the communication network, etc. This variety limits portability of software among these robot skins. In this article, a middleware is proposed that is capable of interacting in principle with any robot skin, through the use of simple drivers. Primarily, the middleware acquires data in real-time and provides its applications with those data in an abstract structure. As a result, the portability of algorithms implemented for large-scale tactile data processing is greatly increased among various available and future robot skins.**

I. Introduction

Tactile sensors have been extensively researched since 1980 [1]. However, only recent research activities in physical human-robot interaction [2] have stressed the importance of the use of large-scale tactile systems, specifically in the form of robot skins [3].

When considering a robot skin, possibly covering large robot body parts, novel issues arise at the design level, such as scalability [4], conformance [5], wiring [6], and networking [7], just to name but few. During the past few years, many examples of robot skin systems have been presented in the literature, either based on modular designs, for example in [5], [8], or on non-modular but flexible and highly conformable sensor arrays, for example in [9], [10]. As far as communication and networking are concerned, different solutions have been investigated, such as PCI bus [11], CAN bus [5], SMBus [12], EtherCAT [7], UART bus [8], etc. However, at the computational level, it is necessary to devise strategies for data acquisition and processing, which is the focus of this article. To the best of the authors' knowledge, before the publication of the authors' earlier work [13], the necessity for a middleware is only mentioned in [14], where a method for organizing tactile data originating from a sensory suit is presented, which is specific only to the task at hand.

As a matter of fact, huge amounts of tactile data originating from distributed sources are of limited use if not processed in time and according to well-defined contact models [15] to extract meaningful information, e.g., force distribution. At a first glance, it may be argued that tactile data processing can be treated using techniques borrowed from Computer Vision. However, there is no shortage of reasons to think that these two fields are different.

- Tactile elements (*taxels* in short) are distributed over robot body parts with varying shape and curvature. As a consequence, taxel locations does not form a square regular grid. This renders such concepts as *bitmaps* invalid, as they assume data to be present on a regular and well-defined 2D grid. On the contrary, *tactile images* are hardly structured [16] in a regular arrangement.
- The actual tactile image depends on specific robot-environment configurations [17]. Contacts may arise as a consequence of robot postures originating from control actions. Moreover, such interaction phenomena as *self-touch* need to be considered since the tactile image strictly depends on the robot configuration only.
- Different robot body parts may demand different requirements from a robot skin in terms of density, resolution or sensitivity of the tactile sensors. Unlike bitmaps, the tactile image needs to be processed taking into account the peculiar characteristics of the body area which originates the tactile data.
- In analogy with biological skin, robot skin may be designed to contain different types of transducers, whose values may be correlated. At a computational level, different tactile images referring to different sensing modes may need to be processed jointly, e.g., to acquire a better picture of the contact event [8].

In order to handle the acquisition and the efficient processing of such data, the need for a *robot skin middleware* arises, i.e., a software framework capable of providing:

- data access mechanisms such as *best effort*, *periodic* and *event-based* in real-time;
- data abstraction features for high-level information processing tasks, specifically with respect to the robot skin hardware and mechanical implementation, the robot kinematic configuration and the specific robot body part.

The development of software frameworks for Robotics is topic of active research. Relevant examples of such frameworks include OROCOS [18], CLARAty [19], YARP [20] and ROS [21]. However, these frameworks provide general-purpose mechanisms to design and implement concurrent and distributed robot software components. On the contrary, the proposed middleware as a whole, which will be henceforth referred to as *Skinware*, can be used as a

All the authors are with the Department of Informatics, Bioengineering, Robotics and Systems Engineering, University of Genoa, Via Opera Pia 13, 16145, Genoa, Italy.

[1]Corresponding author's e-mail address:
fulvio.mastrogiovanni@unige.it

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n. 231500/ROBOSKIN.

specialized component in any of these frameworks, none of which provide any comparable alternative. It is noteworthy that although the same term *middleware* is used to both refer to the proposed software framework and the aforementioned Robotics software frameworks, Skinware is placed at a layer closer to hardware than that of those frameworks and does not provide an alternative to them.

The main contribution of the article is two-fold.

- The definition of a software architecture able to manage large-scale robot skin in real-time and to provide users with data structures and communication protocols that are independent from the actual robot skin hardware and robot configuration.
- An account of the choices that have been made to implement and validate the defined software framework in real-world scenarios.

The article is organized as follows. Section II presents the terminology associated with the Skinware software framework. Section III gives an overview of the architecture of Skinware. Tests performed on Skinware are presented in Section IV. Conclusion follows.

II. Terminology

A. The Robot Skin Hardware

Adopting a modular design, the robot skin is considered to be composed of the following components. In this article we refer to the ROBOSKIN technology described in [5] (see Figure 1).

- *Taxel*. A single tactile transducer, i.e., a discrete sensing point on the robot surface.
- *Module*. A set of taxels closely located, usually managed by local hardware interfaces
- *Patch*. A network of modules; A patch controls corresponding modules, obtains tactile data and processes them using a more complicated hardware.
- *Network*. The infrastructure connecting patches in the robot skin.

A modular design makes the robot skin more easily scalable and robust. Nevertheless, a non-modular robot skin can be viewed as having one module per taxel and therefore is still compatible with Skinware. As a consequence of such fundamental differences between different robot skin technologies, it is important for the middleware to provide abstractions to reason at the most possible general level in addition to bridges that map these high level abstractions to the real hardware. This is important because design and development of algorithms able to process tactile data independently from their particular hardware implementation is desired.

B. Abstract Concepts

Skinware introduces abstract elements in the robot skin and reasons at their level. These entities also help the end-user view the robot skin from a more abstract perspective and therefore contribute to the portability of the implemented software.

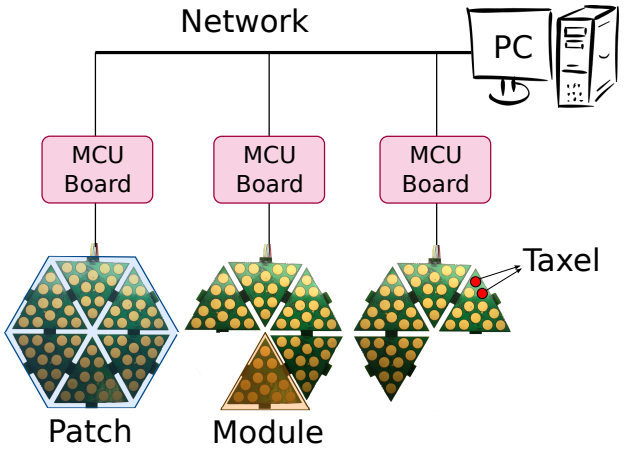

Fig. 1. Taxels, modules, patches and the network in the considered robot skin technology.

- *Sensor*. Although current robot skin implementations are mostly designed to take only pressure information into account, it is possible to design more general robot skins containing other sensor modes, e.g., temperature or vibration [8]. Henceforth, the term sensor will be used regardless of the considered sensing mode.
- *Sensor Type/Layer*. Similar to how the human skin hosts different types of sensors in a layered fashion [22], it is provisioned in Skinware for the underlying robot skin to have multiple sensor types located in multiple layers. The terms *sensor type*, *sensor layer* and simply *layer* refer to the same entity. Each *network* of the robot skin may carry information from multiple layers.
- *Region*. A region is an area-of-interest over the robot skin. This abstract entity represents a set of sensors in the robot skin that need to be grouped for any purpose the end-user may have. Example of such regions include *hand palm*, *forearm* and *manipulators*. The regions may also overlap if the same set of sensors belong to two different conceptual groups, e.g., *forearm* and *robot front*.

Figure 2 shows how it is possible to map these general concepts to virtually any hardware implementation, e.g., to the taxel, module, patch and network structure of the considered robot skin technology.

III. Architecture

Skinware is designed as a layered architecture (Figure 3). Two software layers are always present between the robot skin hardware and user applications, namely the *driver handler* layer (responsible for hardware management and communication) and the *user handler* layer (in charge of communicating with user applications). Furthermore, multiple instances of another layer, namely the *service handler* layer (i.e., for low-level software services), can be optionally present between the driver and the user handler layers, but in a different data path. The main components introduced by Skinware are as follows.

- *Driver*. A software module that is responsible for

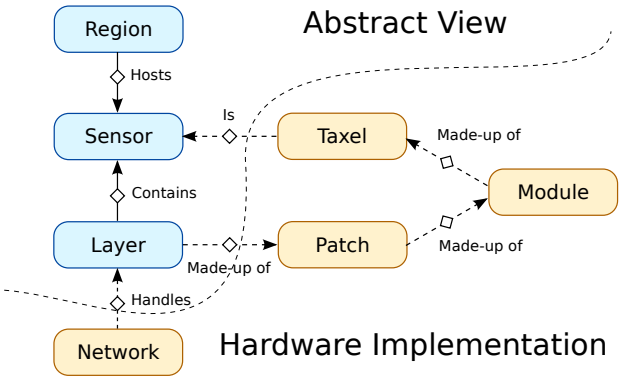

Fig. 2. Mapping of considered robot skin technology to abstract entities provided by Skinware.

communicating both with a network in the robot skin and Skinware. Skinware is capable of communicating with multiple drivers simultaneously, obtaining data from different sensor types connected with different networks.

- *Application*. The terms *application* and *user* both refer to a software module acquiring tactile data through Skinware.
- *Service*. A *user* of sensor data that provides processed data to other users. Generated processed data are propagated with the help of Skinware to other users.
- *Writer*. A real-time thread in Skinware that is responsible for acquiring data from the hardware (through a driver) and *writing* them to internal buffers.
- *Reader*. A real-time thread in Skinware that *reads* the data from internal buffers and places them at user's disposal.
- *Data Frame*. A set of data acquired from sensors of a certain layer in one acquisition period. Each data frame forms a snapshot of the activity in the layer.

In this architecture, the driver handler is closest to the hardware, interacting with the drivers, while the user handler provides data to the applications. The service handler interacts with both standalone services and application-based services. All three components interact using shared locking mechanisms and write/read data to/from shared memory.

This architecture is specifically designed to fulfill the following properties.

- *Ceaseless acquisition*. Skinware must at all times be able to perform data acquisition. This means that *writers* must never be blocked due to the synchronization mechanism with the *readers*.
- *Independence from users*. Data acquisition in Skinware must remain sound and unblocked regardless of the existence of *readers* or their number. This property guarantees predictable behavior in systems where the number of *readers* could dynamically change.

The Skinware architecture utilizes simple network drivers to handle one or more sensor layers. They provide Skinware with layer topology and can be started and stopped dynamically, e.g., for power saving or due to a fault. Corresponding

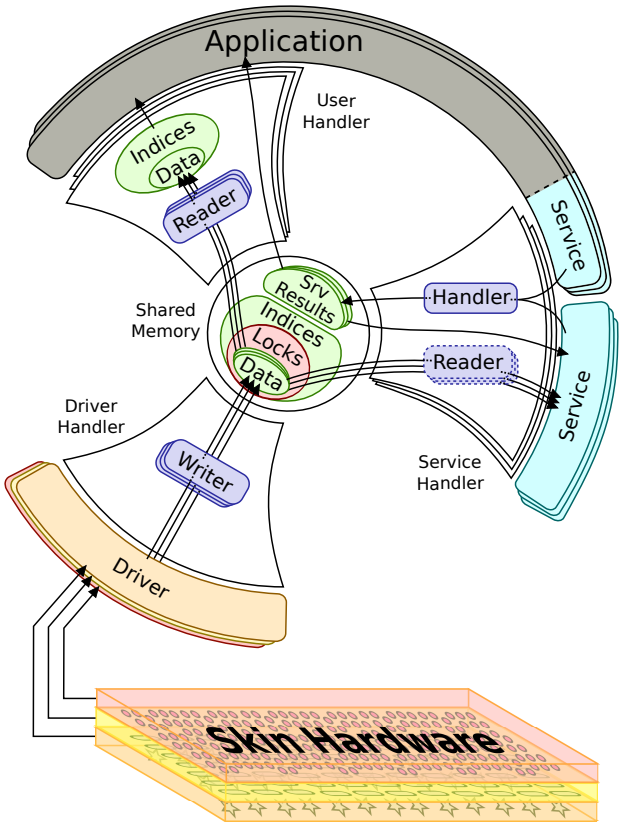

Fig. 3. The architecture of Skinware. Unprocessed data from hardware to applications pass through two layers while processed data reach the same application passing through one extra layer.

to each sensor layer, a writer thread is spawned. This greatly increases robustness and allows for graceful degradation, as a fault in one driver doesn't hinder work of others. The writers in turn write tactile data to shared buffers, utilizing multiple buffers to store data frames in multiple versions.

The applications in turn spawn readers through Skinware that handle synchronization with writers and read tactile data from the most recent version available. Based on the application needs, a reader may acquire data as soon as it is available, periodically or upon application request.

Services, whether standalone or as a side-effect of an application, provide data that are stored in shared memory and are accessed by other services or applications with similar reader-writer interaction. Support for services enables both code and data reuse.

Additional to the tactile data themselves, Skinware supports useful indices on the data made available to applications and services. These indices include the hardware-implied hierarchy (sensor layers, patches, modules and sensors), the abstract region-sensor relation, 2D transformation map of the robot skin [16] and sensor vicinity graph.

IV. Experimental Results

The tests have been performed on a PC with the following specifications.

- Operating System: Ubuntu 12.04 using the Linux kernel 2.6.35.9 patched with RTAI 3.9.

TABLE I

EXECUTION TIMES OF THE WRITERS AND READERS OF BOTH THE CAN LAYER (180 SENSORS) AND THE ETHERCAT LAYER (708 SENSORS).

	Average (ns)	Worst-Case (ns)	STD
$writer_{EC}$	23915	87315	4381
$writer_{CAN}$	20108	44077	2951
$reader_{EC}$	10996	86635	3533
$reader_{CAN}$	6703	47102	1733

- Processor: Intel Core 2 Duo E8200 @2.66GHz.
- RAM: 4GB at 667MHz.
- Ethernet board: Realtek RTL-8169 Gigabit Ethernet.
- CAN board: PEAK-System Technik's PCAN-PCI CAN-Bus controller.

A. Test with the ROBOSKIN Technology

The piece of robot skin used in the first test, shown in Figure 4, consists of 5 patches with the following configuration.

- One patch consisting of 15 modules, placed in a row on the upper side of the saddle. This patch is connected to the PC with a CAN network.
- One patch consisting of 12 modules, one of which was broken during construction and placed on the right side of the saddle, connected with an EtherCAT network.
- Three patches consisting of 16 modules each, connected to the same EtherCAT network, consisting of two EtherCAT slaves.

Therefore, a total of 888 sensors (all taxels) in 74 modules are visible to Skinware. All patches of this robot skin are of the same technology, and therefore of the same type. However, since they are connected through two different networks, and therefore handled by two drivers, they are considered to be in two layers. The setup of this test can be seen in Figure 4.

A visualization program has been implemented and included with Skinware. A simple motion detection service has also been implemented that identifies direction of sliding motion on the robot skin. This test has been accompanied by a script that periodically checks the outputs of Skinware for device drivers that have reported error and tries removing and reloading them. To better observe this process, the visualization program highlights layers that are not working.

During the execution of the test, the CAN or the EtherCAT network cables have been randomly detached and later reattached. Figure 5 shows the continued execution of the visualization program and the motion detection service working uninterrupted while the CAN cable is removed. Once reattached, Skinware continues acquisition from the newly restarted driver and the results are visible in the visualization program.

This test demonstrates the robustness of Skinware and the simplicity of recovery from unexpected errors. During one hour execution of this test, execution time data has been gathered from the two writers as well as the two readers. Table I shows information on execution times for both layers.

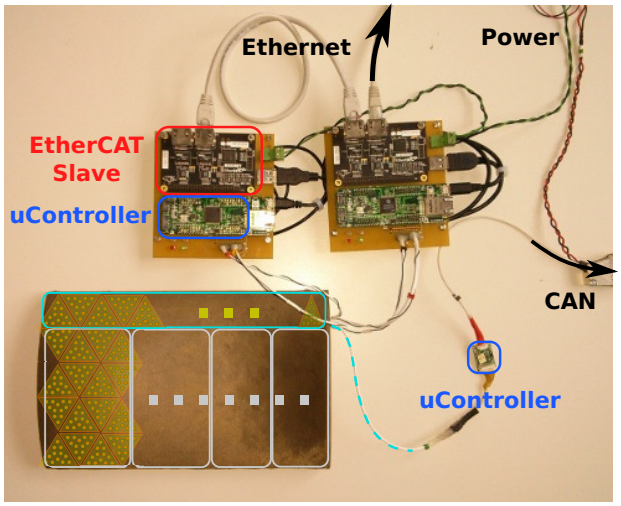

Fig. 4. The test hardware shaped like a saddle. The 5 patches underneath an elastomer are indicated by cyan (connected to CAN) and gray (connected to EtherCAT). In this image, the microcontroller interfacing with the CAN network and two microcontrollers connected to EtherCAT slaves are also visible. The approximate layout of the modules and patches on the saddle are overlaid using an image manipulation software.

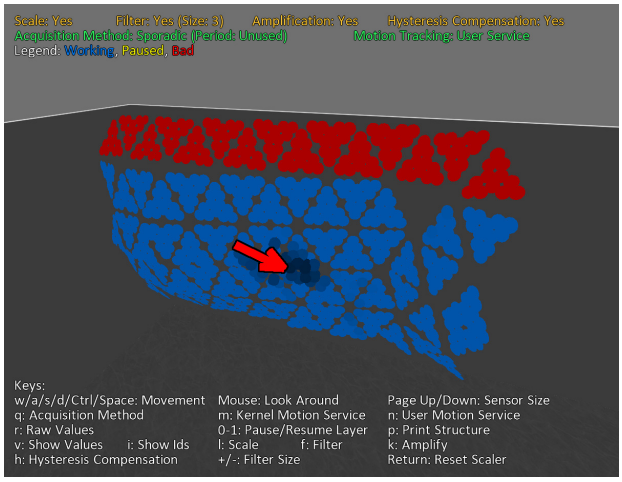

Fig. 5. The visualization program included with Skinware. This program is showing tactile data received from the robot skin in Figure 4. Additionally, it is attached to the motion tracking service and is visualizing its result. In this screen shot, the CAN cable is disconnected. The sensors attached by the EtherCAT network are still functional. Once the cable is reattached, the sensors attached by the CAN network will continue producing responses. The texts are guides in the interface and are irrelevant to this article.

B. Performance Test

Additional to the experimental test, a virtual driver, simulating a robot skin with 30848 sensors divided across multiple layers has been implemented with which the Skinware has been tested. 30848 sensors roughly translates to $1m^2$ of robot skin with the technology in [5]. Studies in [23] have shown this to be a reasonable coverage area for a humanoid body.

$$\#_{sensors} = \frac{1m^2}{\underbrace{3.89cm^2}_{module\ area}} \times \underbrace{12}_{sensor\ per\ module} \simeq 30848$$

The tests with the virtual driver have been performed extensively, measuring the performance of *readers* and *writers* under light and heavy load. The effect of number of sensor layers on the performance of Skinware is simultaneously studied. In these tests, the virtual skin, consisting of 30848 sensors has the following characteristics.

- There are 12 sensors per module.
- There are 4 to 16 modules per patch.
- The 30848 sensors are divided equally among the layers.
- Each *writer* has a random acquisition rate of between 30Hz and 50Hz.
- Each *reader* also has a random and likely different acquisition rate between 10Hz and 100Hz.

The tests have been performed with varying number of sensor layers and user applications. Each configuration itself has been tested multiple times and the relevant data aggregated, resulting in an overall number of about 4000 test executions. The number of sensor layers are varied from one to five layers, having the sensors equally divided among them. The number of user applications are varied from one to fifteen.

Figure 6a shows worst-case execution time of *writers* under different configurations while Figure 6b shows worst-case execution time of *readers*. It can be observed from these figures that, as the number of layers increase, the overall execution time of the *writers* and the *readers* also increase, even though for smaller number of user applications this change is smaller. It can be observed that these values change more slowly for higher number of layers. It can also be observed that as the number of user applications increase, the execution time of the *writers* and the *readers* also increase.

These observed behaviors can be explained by increasing cost of synchronization in the presence of higher number of layers and higher number of user applications. The number of layers influences the number of concurrent *writers* and *readers*. As the number of sensor layers increases, the overall cost of sensor data acquisition remains constant (since the number of sensors is constant), but the synchronization between each *writer* and its corresponding *readers* is executed multiple times (since the number of pairs of *writers* and corresponding set of *readers* increases).

The increase in execution time is minor when the number of user applications is small because synchronization with a fewer number of *readers* is less costly. The detailed explanation of this behavior is out of the scope of this article. The influence of the number of user applications on the execution times is due to the same increased cost of synchronization with multiple readers.

It is worth noting that the execution times have been measured by a wall-clock time due to the lack of precise execution time calculation in the underlying real-time system. This implies that actual time spent by the *writers* and the *readers* is indeed smaller than what is reported in Figure 6. An Increased number of user application implies an increased number of threads, which on the dual-core test PC, implies a higher chance of task preemption. Therefore, the increased execution time with higher number of user application is partly due to the wall-clock time taking into account the times when the thread spends blocked in the round-robin scheduling queue.

V. Discussion

Skinware emphasizes real-time behavior. This aspect is one that is not present in most common robot software platforms. Yet, as robotic systems evolve, real-time behavior will doubtlessly be a major common ground between all robotic applications. Currently, Skinware is based on RTAI [24]. However, support for other real-time systems is in progress. Consistent worst-case execution times as shown in Figure 6, i.e., without spikes or other such irregularities, indicates that Skinware has been successful in meeting this requirement.

The highly concurrent nature of Skinware requires the utmost attention to thread synchronization. This aspect has been carefully designed, developed and tested to ensure coherency and consistency of data for the end user while incurring minimum overhead. The tests presented above have shown that the implemented synchronization algorithms have tolerable effect on the acquisition process.

There have been three types of applications foreseen by Skinware. Time-critical applications may request readers that acquire data as soon as they are available, periodic applications acquire data at a desired rate, and aperiodic applications acquire them upon request. The experimental test above has been done using aperiodic readers due to the non-real-time nature of a GUI application, while the performance tests were done as periodic applications.

The usage of the sample motion detection service in the experiment above shows how Skinware facilitates integration of processed results from different applications. A service may be standalone, in the sense that its sole purpose is to provide processed data to others, or application-based. Standalone services are fast, specialized or data-independent, such as those generating reflexive responses, interested solely in a particular layer or region, or watchdogs. Application-based services on the other hand, are side effects of the applications that choose to share their computation results with other possible users, e.g., an application detecting texture in contact with finger tips primarily produces these data for itself, but is also able to share the data with other applications through a service.

Overall, the tests presented above have shown the practicality of Skinware in real world situations. It can be observed that the latency introduced by Skinware for small regions of robot skins is below 100 microseconds in the worst case and for the whole humanoid-size robot skin, this worst-case latency is below few milliseconds even under heavy usage.

VI. Conclusions

This article introduces Skinware, a software middleware for the real-time management of large-scale tactile information. On the one hand, the software architecture is able to provide the end-user with a hardware-independent perspective of tactile information. On the other hand, it constitutes

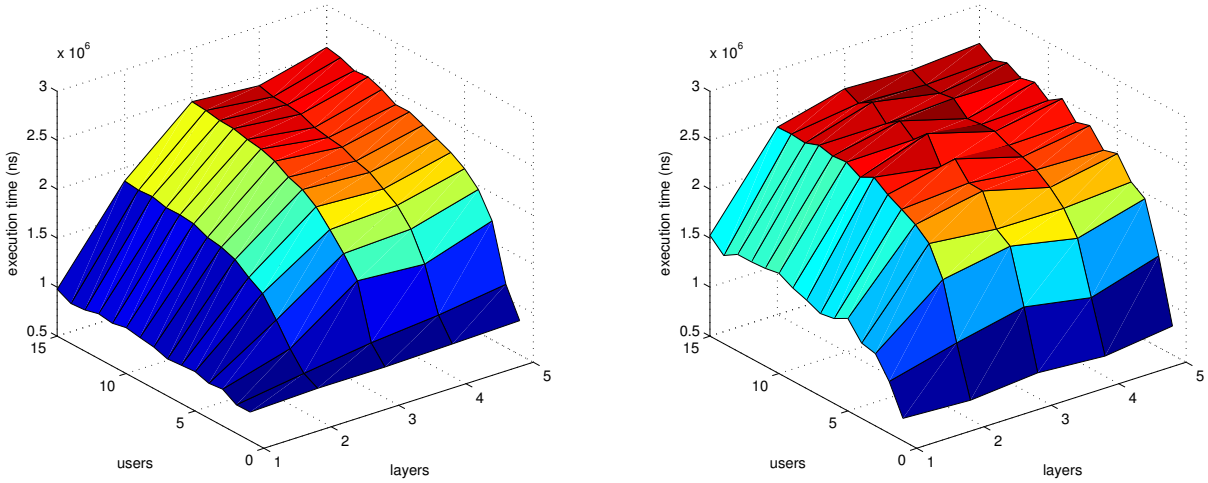

Fig. 6. Worst-case total execution time of *writers* (left) and *readers* of each application (right) when the sensors are divided between 1 to 5 layers, in the presence of 1 to 15 user applications.

the basis for the implementation of higher-level algorithms for tactile data processing, such as those described in [25]. The whole framework is going to be released open source and under a free license.

REFERENCES

[1] M. Lee and H. Nicholls, "Review Article Tactile Sensing for Mechatronics - a State of the Art Survey," *Mechatronics*, vol. 9, no. 1, pp. 1–31, 1999.

[2] B. Argall and A. Billard, "A survey of Tactile Human-Robot Interactions," *Robotics and Autonomous Systems*, vol. 58, no. 10, pp. 1159–1176, 2010.

[3] R. Dahiya, G. Metta, M. Valle, and G. Sandini, "Tactile Sensing - From Humans to Humanoids," *Robotics, IEEE Transactions on*, vol. 26, no. 1, pp. 1–20, 2010.

[4] A. Schmitz, P. Maiolino, M. Maggial, L. Natale, G. Cannata, and G. Metta, "Methods and Technologies for the Implementation of Large-Scale Robot Tactile Sensors," *Robotics, IEEE Transactions on*, vol. 27, no. 3, pp. 389–400, 2011.

[5] G. Cannata, M. Maggiali, G. Metta, and G. Sandini, "An Embedded Artificial Skin for Humanoid Robots," in *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2008 IEEE International Conference on*, 2008, p. 434.

[6] M. Nilsson, "Tactile Sensing with Minimal Wiring Complexity," in *Robotics and Automation, 1999. Proceedings of the 1999 IEEE International Conference on*, vol. 1, 1999, p. 293.

[7] E. Baglini, G. Gannata, and F. Mastrogiovanni, "Design of an Embedded Networking Infrastructure for whole-Body Tactile Sensing in Humanoid Robots," in *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, 2010, p. 671.

[8] P. Mittendorfer and G. Cheng, "Humanoid Multimodal Tactile-Sensing Modules," *Robotics, IEEE Transactions on*, vol. 27, no. 3, pp. 401–410, 2011.

[9] J. Ulmen and M. Cutkosky, "A Robust, Low-Cost and Low-Noise Artificial Skin for Human-Friendly Robots," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 4836–4841.

[10] D. Tawil, D. Rye, and M. Velonaki, "Touch Modality Interpretation for an EIT-based Sensitive Skin," in *Robotics and Automationi (ICRA), 2011. Proceedings of the IEEE International Conference on*, Shanghai, China, 2011.

[11] H. Kawasaki, T. Komatsu, and K. Uchiyama, "Dexterous Anthropomorphic Robot Hand with Distributed Tactile Sensor: Gifu Hand II," *Mechatronics, IEEE/ASME Transactions on*, vol. 7, no. 3, pp. 296–303, 2002.

[12] Y. Ohmura, Y. Kuniyoshi, and A. Nagakubo, "Conformable and scalable tactile sensor skin for curved surfaces," in *Robotics and Automation (ICRA), 2006. Proceedings of the IEEE International Conference on*, 2006, p. 1348.

[13] S. Youssefi, S. Denei, F. Mastrogiovanni, and G. Cannata, "A Middleware for Whole Body Skin-like Tactile Systems," in *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, 2011, pp. 159–164.

[14] Y. Hoshino, M. Inaba, and H. Inoue, "Model and Processing of Whole-body Tactile Sensor Suit for Human-Robot Contact Interaction," in *Robotics and Automation, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 3, 1998, p. 2281.

[15] K. Johnson, *Contact Mechanics*. Cambridge, UK: Cambridge University Press, 1985.

[16] G. Cannata, S. Denei, and F. Mastrogiovanni, "On Internal Models for Representing Tactile Information," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 1108–1113.

[17] ——, "A Framework for Representing Interaction Tasks based on Tactile Data," in *RO-MAN, 2010 IEEE*, 2010, p. 698.

[18] H. Bruyninckx, "Open Robot Control Software: the OROCOS Project," in *Robotics and Automation (ICRA), 2001. Proceedings of the IEEE International Conference on*, vol. 3, 2001, pp. 2523–2528.

[19] R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras, and H. Das, "The CLARAty Architecture for Robotic Autonomy," in *Aerospace Conference, 2001, IEEE Proceedings.*, vol. 1, 2001, pp. 1/121–1/132.

[20] G. Metta, P. Fitzpatrick, and L. Natale, "YARP: Yet Another Robotic Platform," *International Journal of Advanced Robotics Systems*, vol. 3, no. 1, 2006.

[21] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an Open-source Robot Operating System," in *Robotics and Automation (ICRA) Workshop on Open Source Robotics, 2009. Proceedings of the IEEE International Conference on*, Kobe, Japan, May 2009.

[22] J. Schwartz, E. Kandel, and T. Jessell, *Principles of Neural Science*. Prentice-Hall International, 1991.

[23] C.-Y. Yu, C.-H. Lin, and Y.-H. Yang, "Human Body Surface Area Database and Estimation Formula," *Burns : journal of the International Society for Burn Injuries*, vol. 36, no. 5, pp. 616–629, 2010.

[24] P. Mantegazza, E. Dozio, and S. Papacharalambous, "RTAI: Real Time Application Interface," *Linux Journal*, vol. 2000, no. 72es, 2000.

[25] L. Muscari, F. Mastrogiovanni, L. Seminara, M. Capurro, G. Cannata, and M. Valle, "Real-time Reconstruction of Contact Shapes for Large Area Robot Skin," in *Robotics and Automation (ICRA), 2013 Proceedings of the IEEE International Conference on*, 2013.