Development of an Eye- and Gaze-Tracking Mechanism in an Active and Assisted Living Ecosystem

A. Liu Cheng[a,b], N. Llorca[b,c], and G. Latorre[d]

[a]Faculty of Architecture and the Built Environment, Delft University of Technology, Delft, The Netherlands
[b]Facultad de Arquitectura e Ingenierías, Universidad Internacional SEK, Quito, Ecuador
[c]Escuela de Arquitectura, Universidad de Alcalá, Madrid, Spain
[d]Centro de Educación Continua, Escuela Politécnica Nacional, Quito, Ecuador
E-mail: a.liucheng@tudelft.nl, nestor.llorca.arq@uisek.edu.ec, galoget.latorre@epn.edu.ec

Abstract –

This paper details the development of an open-source eye- and gaze-tracking mechanism designed for open, scalable, and decentralized Active and Assisted Living (AAL) ecosystems built on Wireless Sensor and Actuator Networks (WSANs). Said mechanism is deliberately conceived as yet another service-feature in an on-going implementation of an extended intelligent built-environment framework, one motivated and informed by both Information and Communication Technologies (ICTs) as well as by emerging Architecture, Engineering, and Construction (AEC) considerations. It is nevertheless designed as a compatible and subsumable service-feature for existing above-characterized AAL frameworks. The eye- and gaze-tracking mechanism enables the user (1) to engage (i.e., open, shut, slide, turn-on/-off, etc.) with a variety of actuable objects and systems deployed within an intelligent built-environment via sight-enabled identification, selection, and confirmation; and (2) to extract and display personal identity information from recognized familiar faces viewed by the user. The first feature is intended principally (although not exclusively) for users with limited mobility, with the intention to support independence with respect to the control of remotely actuable mechanisms within the built-environment. The second feature is intended to compensate for loss of memory and/or visual acuity associated principally (although not exclusively) with the natural aging process. As with previously developed service-features, the present mechanism intends to increase the quality of life of its user(s) in an affordable, intuitive, and highly intelligent manner.

Keywords –

Intelligent Built-Environments, Active and Assisted Living, Wireless Sensor and Actuator Network, Internet of Things, Adaptive Architecture

1 Introduction

The eye- and gaze-tracking mechanism detailed in this paper is situated within the *Active and Assisted Living* / *Ambient Assisted Living* (AAL) discourse [1–4]. Its objective is to enable the user (1) to open, shut, slide actuable windows and doors as well as to turn lights on or off within an intelligent built-environment via sight-enabled identification, selection, and execution; and (2) to extract and display personal identity information from recognized familiar faces viewed by the user. Although this mechanism is designed to be compatible with existing AAL or AAL-pertinent ecosystems and frameworks (e.g., *CASAS* [5], *The Aware Home* [6], *PlaceLab* [7]; and more recently in the last three years: *eServices* [8], *REACH* [9], *WITS* [10], etc.), it is deliberately designed as a service-feature in an ongoing implementation of an intelligent built-environment framework inspired by what Oosterhuis has called a *Society of Home*, where users and their surrounding objects / systems communicate and thereby instantiate a home of *Internet of Things and People*; as well as a *Society of Building Components*, where the intelligent built-environment's objects / systems act, react, and interact computationally as well as physically [11] towards user well-being. In order to realize this *Society of Home*, emerging trends in *Architecture, Engineering, and Construction* (AEC) research laboratories in academia and industry—such us robotic fabrication, computational design and optimization, etc.—are considered in conjunction with *Information and Communication Technologies* (ICTs) as well as allied and resulting technical services that typically dominate the AAL discourse. This dual consideration, while adopted from the early stages of conception and design, ascertains mutual complementarity [12] between ICTs and AEC features where neither requires retrofitting from or to the other. In such an intelligent built-environment, both ICTs and AECs features serve to enhance—in conjunction—the user's quality of life.

The eye- and gaze-tracking mechanism's first objective / feature intends principally (although not exclusively) to assists users with limited mobility to control remotely actuable mechanisms within the built-environment. The second objective / feature intends to compensate for the user's loss of memory and/or visual acuity—with respect to recognition of familiar faces—associated principally (although not exclusively) with the natural aging process. That is to say, these two features serve as extensions of the user's physical and mental capabilities (with respect to the described contexts and scopes) in a way that enables the user to retain a degree of independence otherwise impossible without technical and technological assistance. This is in line with current trends where *Computer Vision* is integrated into assistive services and solutions (see, for example, [13]).

The present mechanism is deployed within an inherited *Wireless Sensor and Actuator Network* (WSAN) [14]. This WSAN's ecosystem is highly heterogenous in architecture, communication protocols, and network topology. Some of its nodes are embedded in the physical environment, while others are embedded in ambulant systems or in wearable devices. Capitalizing on the different levels of integration within the built-environment, all nodes work in conjunction to instantiate and to sustain a variety of intelligent service-features. Some service-features are strictly computational in character, while others involve computation and physical action, reaction, interaction as input or output; some are completely localized (i.e., dependent strictly and solely on the local network) while others are either hybrid (i.e., relying on services present both in the local network as well as in distributed cloud-based services) or based entirely on free cloud-based services via their corresponding *Application Program Interface*s (APIs). The hardware of the eye- and gaze-tracking mechanism is embedded in a wearable device, which works together with architecture-embedded nodes and systems, as well as with a variety of communication protocols, to instantiate the above-mentioned features. In order to center on the discussed mechanism, the systems and subsystems implemented in the inherited WSAN are presupposed without delving into their description and functionality details.

This paper consists of four main sections. Section 2 details the *Concept and Approach*, where the system is described and the implementation scope is defined. Section 3 explains the *Methodology and Implementation*, where the software and hardware implementations are described and the sequence of operation is detailed. Finally, Section 4 presents the *Results* that validate the present *proof-of-concept* implementation and provides a discussion on limitations, further work, and conclusions.

2 Concept and Approach

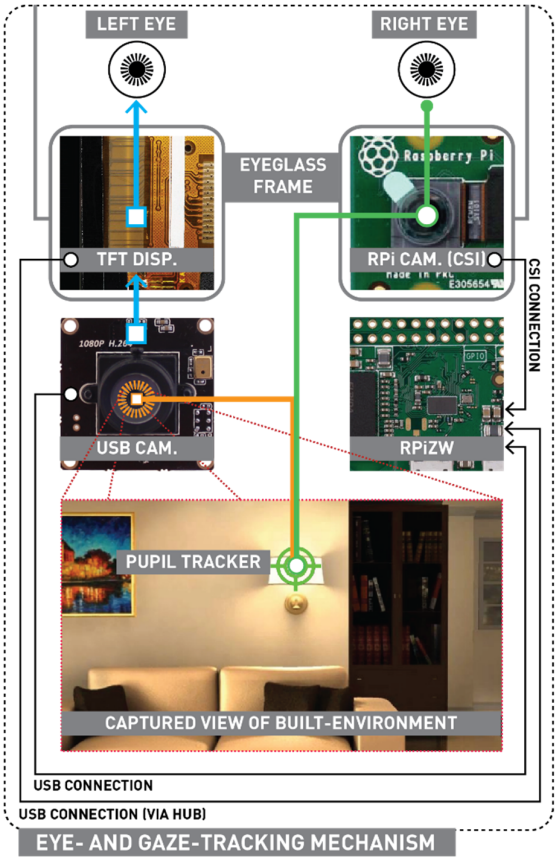

Figure 1. Concept and System Architecture of the Eye- and Gaze-Tracking Mechanism.

The eye- and gaze-tracking mechanism integrates the following hardware components into a generic eyeglass frame: a (i) Raspberry Pi Zero W (RPiZW); (ii) generic USB TFT Display; (iii) generic USB Camera (USBCam); (iv) and a CSI Raspberry Pi Camera v. 2.1. (RPiCam). The RPiCam is used to track the movement of the pupil corresponding to the right eye, while the USBCam captures the environment viewed and projects this scene back to the left eye via the TFT Display with a real-time update frequency. The position of the pupil is mapped on top of the captured scene, and gaze-tracking is enabled by analyzing the position of the pupil—and its movements—in relation with recognizable objects captured in the scene (see Figure 1). The following recognizable objects are presently defined: a (1) sofa, (2) wall-light, (3) window, (4) ceiling-light, (5) bookcase, (6) lamp, (7) wall-art, and (8) a dining table (see Figure 2, *Bottom*). This represents a generic sampling of objects contained in a given built-environment, where some actuate and others do not.

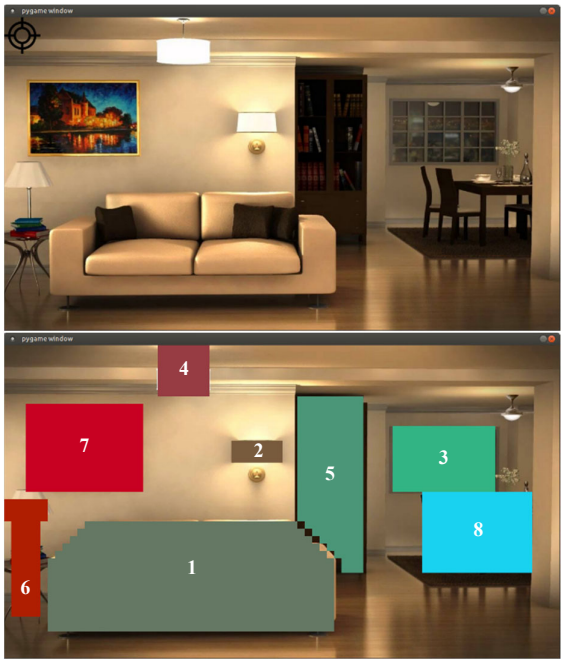

Figure 2. Top: Captured-gaze of generic sample built-environment. Bottom: Object-boundaries overlaid on top of objects represented. Both generated with PyGame.

In this implementation, the engagement with actuable objects and systems (e.g., doors, windows, lights, etc.) is limited to their generic virtual representation (see Figure 2, *Top*). That is, while previous implementations have required the building of physical representations in real-scale, the present development only requires confirmation of engagement at a software level to ascertain its functionality. Accordingly, a black-box approach is adopted with respect to the other subsystems of the inherited WSAN. In this manner, the present mechanism is said to actuate a window, a door; or to turn on/off a light, etc., if the user is detected to be engaging with a given object / system and the eye- and gaze-tracking mechanism does indeed respond by sending a signal to actuate or to turn on/off.

A caveat on the architecture / built-environment used as the captured scenes in this implementation: the type of architecture displayed intends to represent an average contemporary built-environment. As such, it is meant as a neutral representation, and not as an illustration of the type of environments to be generated by considering both ICTs and emerging AEC features. Nevertheless, as simple and reduced as representation as it may be, it does serve to illustrate the functionality of the present mechanism in a variety of existing AAL or AAL-compatible built-environments.

As mentioned at the beginning of this section, the position of the tracked pupil is overlaid with frequently updated captured scenes. When the eye-target symbol representing the location of said pupil lands on a non-defined region, its color remains black and no actuation options appear on the display (see Figure 2, *Top*). The eye-target symbol changes color whenever the system detects that it has entered any of the eight defined regions. N.B.: entering a region occurs when the overlap between the eye-target symbol's area and a given object's defined region is greater than 50% of the symbol's overall area—that is, when the majority of the symbol representing the eye is inside a given object's region.

If the eye-target symbol's color turns red (see Figure 3), the console outputs confirmation of recognition of viewed object while warning that no actuations are associated with said object—as Kolarevic points out, an adaptive environment is more appropriately conceived as one containing both high-tech. and low-tech. objects [15], where the former are actuatable while the latter are fixed. If, however, the eye-target symbol's color turns green (see Figure 4, *Wall* and *Wall-Light*), the console confirms object recognition and provides a list of actuation options associated with said object. The user is able to select any of the available actuation options by looking at it (i.e., by moving the eye-target symbol to sit on top of the preferred option) and then blinking twice to confirm selection and trigger actuation execution (i.e., sending a confirmation signal to the responsible node via which the actuatable object is controlled). Finally, if the eye-target symbol's color turns yellow (see Figure 4, *Wall-Art*) the console confirms object recognition and outputs a series of options pertaining not to physical actuation but to other kinds of engagement, namely information finding about the object, etc.—for example, in Figure 4, *Wall-Art*, which represents the user looking at a generic instance of wall-art, options to find information about said art on the web or to take, store, and share a picture of it are outputted by the console.

This first feature concerns the recognition of actuatable and static objects and systems within the built-environment. In addition to this feature, and as mentioned in the *Introduction*, the mechanisms is equipped with a second feature that concerns the extraction of information corresponding to recognized familiar faces. In this second feature, recognized human faces—that is, faces that have already undergone *Machine Learning* (ML)-based training for identity recognition for a given user—are also a *de facto* defined region. If the eye-target symbol enters this region, the console outputs previously stored information about the recognized person (see Figure 5). Since this region is not of the same class as the previously listed eight regions corresponding to the actuatable / static objects

in the present generic built-environment, the eye-target symbol disappears to give way to a rectangular boundary that contains the recognized face. This change of representation explicitly distinguishes identified and viewed non-human and human objects. As illustrated in Figure 5, the probability of recognition of a face is overlaid on top of the captured scene. Accordingly, in this particular figure, the user is informed that the viewed person is "Alejandra" with a 77.98% probability—for simplicity, any probability higher than 75% is accepted as accurate in this implementation.

The console portion of the figure (Figure 5, *Bottom*) shows multiple console outputs, each of which represents the processing of a new instance of a captured scene. In this particular figure, it may be appreciated how the prediction probability fluctuates depending on lighting conditions, distance, etc. Nevertheless, all shown instances correctly and strongly indicate that the person viewed is "Alejandra". Furthermore, in addition to successful prediction probabilities, the console also outputs other useful and pertinent information in order to compensate for loss of memory and/or visual acuity. For example, in this figure, the mechanism informs the user that this is "Alejandra", and that she is: "24 years old"; an "Architecture student"; the user's "daughter-in-law"; and the wife of "Luis Francisco", who may be presumed to be the user's son. All this information pertaining to this familiar face are previously stored for the user and will naturally differ from user to user. In the present setup, only a small set of data (i.e., Full name, Age, Profession, etc.) is provided to illustrate the mechanism's functionality. However, the amount and type of data may be added or removed depending on the user's preference and/or need. The principal purpose of this second feature is to enable to user to always recognize the people in her/his immediate surroundings.

3 Methodology and Implementation

3.1 Eye- and Gaze-Tracking for Object Actuation

The eye- and gaze-tracking mechanism as a means to engage actuatable objects / systems is implemented via two Python programs: (1) *main_eye_detector.py* and (2) *pygame_window.py.* The first (built with *OpenCV* and *Numpy*) is responsible for identifying and tracking the pupil; while the second (built with *Sys* and *PyGame*) is responsible for generating and/or displaying the representation of a given captured built-environment scene. It is in this latter that the regions corresponding to the eight actuatable / static objects are defined; and where the overlapping between eye-target symbol and the defined objects' boundary is calculated.

3.2 Eye- and Gaze-Tracking for Human Identity Recognition

The component of the eye- and gaze-tracking mechanism responsible for human identity recognition depends on the inherited facial-identity and -expression mechanism previously developed by the authors [16]. In the previous work as with the present, the facial identity recognition component is implemented in the local network via Google Brain®'s *TensorFlow*™ [17]. That is, TensorFlow™ is installed in the RPiZW's *Raspbian* operating system and its cloud-based services are implemented via Python. The implementation of this service has two phases. In the first phase, and during execution of its *Multi-task Cascaded Convolutional Networks* (MTCNN) face detection model, the camera captures a given subject's face from different positions, orientations, and angles. All the people to be added to the eye- and gaze-tracking mechanism's users circle of familiar faces must undergo this phase. The second phase is the actual real-time execution of facial recognition, following the successful acquisition of analysed and stored faces from the first phase.

4 Results and Conclusion

With respect to the object actuation feature (see Section 3.1) of the present mechanism, three volunteers (with varying ages) (see *Acknowledgements*) in addition to the authors tested its functionality and performance via the following five steps:

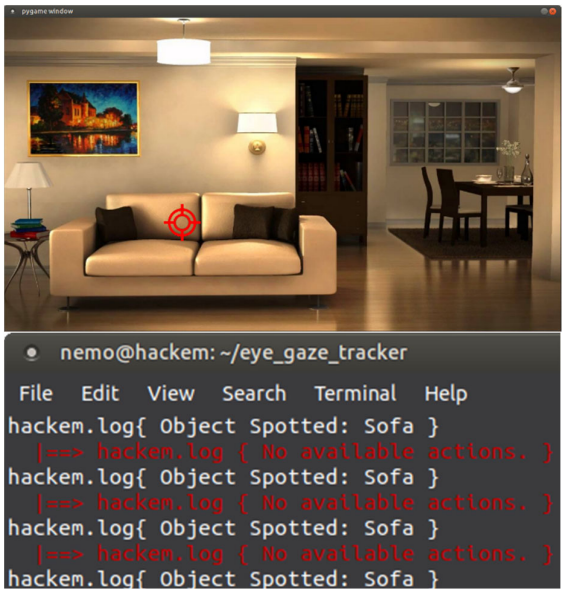

Figure 3. Pupil detected on object *Sofa*, with no available actuations.

1. At initialization of *PyGame*'s represented captured-scene, the eye-target symbol first appears in the upper-left corner. The user then moves her/his pupil in question (i.e., right-eye's pupil) and blinks twice over defined objects in order to first calibrate the eye-tracking component.
2. Once the eye-tracking component is calibrated the user is directed to look at the *window*, at which point the eye-target symbol changes colors to green, and the console first recognizes the actuatable object and the outputs the available actuation options: *open* or *close* (see Figure 4, *Window*). The user selects *open* and a confirmation of a corresponding execution signal sent is ascertained—the same, *mutatis mutandis*, for *close*.
3. The user is then instructed to look at the *sofa*, which is a non-actuating / fixed object that is nevertheless recognized. At this point the eye-target symbol turns red to indicate that no available actuation options exist for this object (see Figure 3).
4. The user's attention is turned to the *wall-art*, which is a non-actuatable object yet is nevertheless an object associated with non-physical actions. That is, after the eye-target symbol turns yellow (to indicate that the object in question is neither actuatable or fixed with no available options), the user is given the choice *to search for more information on the art online* or *to take, store, and share a picture of the art* (see Figure 4, *Wall-Art*). The user is instructed to engage in both, and a corresponding confirmation signal is ascertained.
5. Finally, the user is instructed to look at any of the light fixtures (see Figure 4, *Wall-Light*). Any of these fixtures present the user with two options: *turn on* or *turn off*. The user is instructed to engage in both, sequentially, and corresponding confirmation signals are ascertained.

Again, and as mentioned in Section 2, the present setup has its actuations act on virtual representations of generic objects found in an average built-environment. For the scope of the present implementation, it is only necessary to ascertain that a confirmation signal is sent to a respective enabling-node (enabling of the window, the lights, etc.) in order to demonstrate the successful functionality of the present *proof-of-concept*. In all of the above five steps mentioned, corresponding confirmation signals were indeed ascertained.

In the present setup, only three types of responses / options have been considered: (1) actuatable object (green eye-target); (2) static object (red eye-target); and (3) static object associated with non-physical actions (yellow eye-target). However, other possible responses / options may be envisioned for subsequent iterations of this feature of the mechanism.

Figure 4. Top-to-Bottom: Pupil on object *Window* (green target/text); *Wall-Art* (yellow target/text); and *Wall-Light* (green target/text).

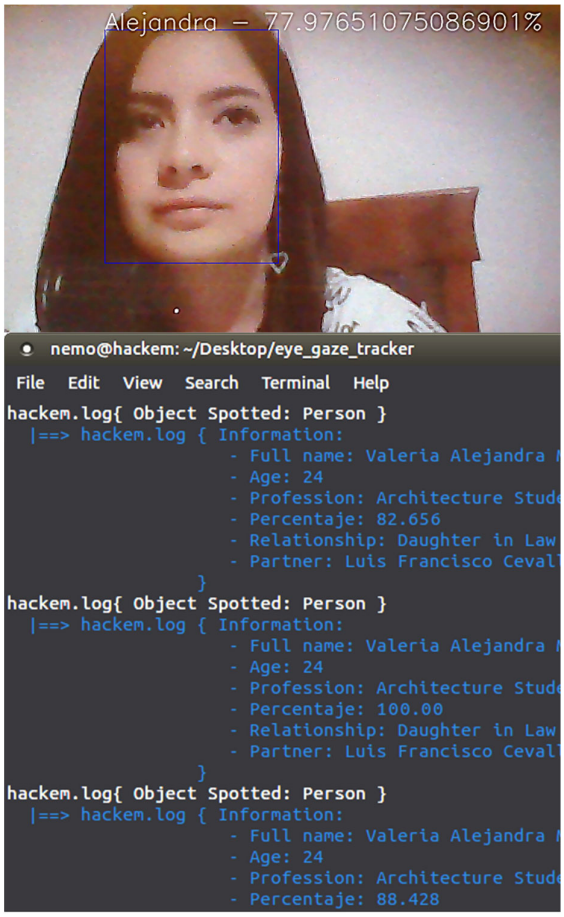

Figure 5. Recognition of a familiar face and corresponding output of associated information pertaining to the identified person.

With respect to the human identity recognition feature (see Section 3.2) of the present mechanism, a database of recognizable faces was first established, followed by a sequence of trials by the same three volunteers and the authors to gauge the feature's human identity recognition capabilities. The feature performed as expected, where the prediction accuracy probabilities were consistently above 70%.

In this paper an eye- and gaze-tracking mechanism has been presented. Said mechanism enables the user (1) to engage (i.e., open, shut, slide, turn-on/-off, etc.) with a variety of actuable objects and systems deployed within an intelligent built-environment via sight-enabled identification, selection, and confirmation; and (2) to extract and display personal identity information from recognized familiar faces viewed by the user. As with previously developed service-features, the present mechanism intends to increase the quality of life of its user(s) in an affordable, intuitive, and highly intelligent manner. Although the present setup is limited and not yet ready for widespread adoption (being in a *Technology Readiness Level* [18] of 5), the detailed *proof-of-concept* implementation's validated functionality and performance indicates further potential for development, which is presently being undertaken. Furthermore, in order to explore facial recognition alternatives in order to enhance the efficiency—with respect to the performance of the local network—of the mechanism, a subsequent version of the eye- and gaze-tracking mechanism is being implemented with *Keras* [19].

Acknowledgement

The authors acknowledge the kind assistance that the following people provided during the implementation and trials of the detailed mechanism: Steffany A. Cevallos G., Nelson B. Solano Y., and Jonathan G. Díaz A.

References

[1] Dimitrievski, A., Zdravevski, E., Lameski, P., and Trajkovik, V. A survey of Ambient Assisted Living systems. Challenges and opportunities. In *Proceedings, 2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP). Cluj-Napoca, Romania, September 8-10, 2016*. IEEE, pages 49–53, Piscataway, NJ, 2016. DOI=10.1109/ICCP.2016.7737121.

[2] Flórez-Revuelta, F. and Chaaraoui, A. A., Eds. Active and Assisted Living: Technologies and Applications. Healthcare technologies series, volume 6. The Institution of Engineering and Technology, Stevenage, UK, 2016.

[3] Grzegorzek, M., Gertych, A., Aumayr, G., and Piętka, E. Trends in Active and Assisted Living - Open hardware architecture, Human Data Interpretation, intervention and assistance. *Computers in biology and medicine*, volume 95: 234–235, 2018.

[4] Byrne, C., Collier, R., and O'Hare, G. A Review and Classification of Assisted Living Systems. *Information*, volume 9, 7: 182, 2018.

[5] Rashidi, P. and Cook, D. J. Keeping the Resident in the Loop. Adapting the Smart Home to the User. *IEEE Trans. Syst., Man, Cybern. A*, volume 39, 5: 949–959, 2009.

[6] Kidd, C. D., Orr, R., Abowd, G. D., Atkeson, C. G., Essa, I. A., MacIntyre, B., Mynatt, E. D., and Starner, T. The Aware Home: A Living Laboratory for Ubiquitous Computing Research.

In *Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*. Springer Verlag, London, UK, 1999.

[7] Intille, S. S., Larson, K., Tapia, E. M., Beaudin, J. S., Kaushik, P., Nawyn, J., and Rockinson, R. Using a Live-In Laboratory for Ubiquitous Computing Research. In *Pervasive Computing*. Springer Berlin Heidelberg, pages 349–365, Berlin, Heidelberg, 2006.

[8] Marcelino, I., Laza, R., Domingues, P., Gómez-Meire, S., Fdez-Riverola, F., and Pereira, A. Active and Assisted Living Ecosystem for the Elderly. *Sensors (Basel, Switzerland)*, volume 18, 4, 2018.

[9] European Union's Horizon 2020 Research and Innovation Programme. REACH: Responsive Engagement of the Elderly promoting Activity and Customized Healthcare. *(Grant agreement No 690425.)*. On-line: http://reach2020.eu. Accessed: 20/04/2017.

[10] Yao, L., Sheng, Q. Z., Benatallah, B., Dustdar, S., Wang, X., Shemshadi, A., and Kanhere, S. S. WITS. An IoT-endowed computational framework for activity recognition in personalized smart homes. *Computing*, volume 100, 4: 369–385, 2018.

[11] Oosterhuis, K., 2014. Caught in the Act. In *ALIVE. Advancements in adaptive architecture*, M. Kretzer and L. Hovestadt, Eds. Applied Virtuality Book Series v.8. Birkhäuser, Basel/Berlin/Boston, pages 114–119.

[12] Milgrom, P. R. The economics of modern manufacturing: technology, strategy, and organization. *The American Economic Review*, volume 80, 3: 511–528, 1990.

[13] Leo, M. and Farinella, G. M., Eds. Computer Vision for Assistive Healthcare. Elsevier, 2018.

[14] Liu Cheng, A. and Bier, H. Extension of a High-Resolution Intelligence Implementation via Design-to-Robotic-Production and -Operation strategies. In *Proceedings of the 35th International Symposium on Automation and Robotics in Construction (ISARC) 2018*, pages 1005–1012, Berlin, Germany, 2018.

[15] Kolarevic, B., 2014. Outlook. Adaptive Architecture: Low-Tech, High-Tech, or Both? In *ALIVE. Advancements in adaptive architecture*, M. Kretzer and L. Hovestadt, Eds. Applied Virtuality Book Series v.8. Birkhäuser, Basel/Berlin/Boston, pages 148–157.

[16] Liu Cheng, A., Bier, H., and Latorre, G. Actuation Confirmation and Negation via Facial-Identity and -Expression Recognition. In *Proceedings of the 3rd IEEE Ecuador Technical Chapters Meeting (ETCM) 2018*, 2018.

[17] TensorFlow™. An open source machine learning framework for everyone, 2018. On-line: https://www.tensorflow.org/. Accessed: 20/04/2018.

[18] European Association of Research and Technology Organisations. The TRL Scale as a Research & Innovation Policy TOOL. *EARTO Recommendations*, 2014. On-line: http://www.earto.eu/fileadmin/content/03_Publications/The_TRL_Scale_as_a_R_I_Policy_Tool_-_EARTO_Recommendations_-_Final.pdf. Accessed: 07/01/2015.

[19] Keras®. Keras: The Python Deep Learning Library, 2019. On-line: https://keras.io/. Accessed: 10/01/2019.