

# A Distributed Architecture for Interacting with NAO

Fabien Badeig, Quentin Pelorson, Soraya Arias, Vincent Drouard, Israel Dejene Gebru, Xiaofei Li, Georgios Evangelidis, Radu Horaud

▶ **To cite this version:**

Fabien Badeig, Quentin Pelorson, Soraya Arias, Vincent Drouard, Israel Dejene Gebru, et al.. A Distributed Architecture for Interacting with NAO. International Conference on Multimodal Interaction, Nov 2015, Seattle, WA, United States. pp.385-386, 10.1145/2818346.2823303 . hal-01201716

**HAL Id: hal-01201716**
**https://inria.hal.science/hal-01201716**

Submitted on 2 Oct 2015

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# A Distributed Architecture for Interacting with NAO *

F. Badeig Q. Pelorson S. Arias V. Drouard
I. D. Gebru X. Li G. Evangelidis R. P. Horaud

INRIA Grenoble Rhone-Alpes
Montbonnot-Saint-Martin, France
fabien.badeig@inria.fr

One of the main applications of the humanoid robot NAO – a small robot companion – is human-robot interaction (HRI). NAO is particularly well suited for HRI applications because of its design, hardware specifications, programming capabilities, and affordable cost. Indeed, NAO can stand up, walk, wander, dance, play soccer, sit down, recognize and grasp simple objects, detect and identify people, localize sounds, understand some spoken words, engage itself in simple and goal-directed dialogs, and synthesize speech. This is made possible due to the robot's 24 degree-of-freedom articulated structure (body, legs, feet, arms, hands, head, etc.), motors, cameras, microphones, etc., as well as to its on-board computing hardware and embedded software, e.g., robot motion control.

Nevertheless, the current NAO configuration has two drawbacks that restrict the complexity of interactive behaviors that could potentially be implemented. Firstly, the on-board computing resources are inherently limited, which implies that it is difficult to implement sophisticated computer vision and audio signal analysis algorithms required by advanced interactive tasks. Secondly, programming new robot functionalities currently implies the development of embedded software, which is a difficult task in its own right necessitating specialized knowledge. The vast majority of HRI practitioners may not have this kind of expertise and hence they cannot easily and quickly implement their ideas, carry out thorough experimental validations, and design proof-of-concept demonstrators.

We have developed a distributed software architecture that attempts to overcome these two limitations. Broadly speaking, NAO's on-board computing resources are augmented with external computing resources. The latter is a computer platform with its CPUs, GPUs, memory, operating system, libraries, software packages, internet access, etc. This configuration enables easy and fast development in Matlab, C, C++, or Python. Moreover, it allows the user to combine on-board libraries (motion control, face detection, etc.) with external toolboxes, e.g., OpenCv.

An overview of the proposed software architecture is shown on Fig. 1. Data coming from NAO (motor positions, images, sampled microphone signals, or data produced by on-board computing modules) are fed into the external computer. Conversely, the latter can control the robot. Currently we have developed three internal-to-external interfaces: *vision*, *audio*, and *locomotion*. The role of these interfaces is twofold: (i) to feed the data into a memory space that is subsequently shared with existing software modules or with modules under development and (ii) to send back to the robot commands that are generated by the external software modules. Although these modules may be developed in a variety of programming languages, special emphasis was put to allow integration with the Matlab programming environment. This allows access to an infinitely large number of libraries and toolboxes. It also allows high-level robot control via Matlab "commands". On-board software modules, e.g., face detection, can be accessed directly from the Matlab environment.

To date, the proposed architecture is demonstrated with five examples of various degrees of complexity:

- *Robot posture control.* This demonstrates that robot actions involving many degrees of freedom can be executed via Matlab commands, e.g., "Stand", "Sit", etc. Matlab programs can easily read the robot-posture data, specify the overall speed of the action, etc. Note that the actuator controller runs on-board and the proposed system allows easy access to this controller, at a high level.

- *Multiple face detection.* This is another on-board software module that can be invoked via the proposed architecture. Note that the robot does not have a display and the developer is not aware of the actual output of vision modules running on-board. This illustrates the integration with C++ modules and libraries.

- *Face detection and robot motion.* This illustrates that several on-board modules can be easily combined together. It involves parallel execution of several modules, possibly written in different programming languages, e.g., a vision module in C++ and a locomotion module in Matlab.

- *Audio-visual localization.* This demonstrates an even more complex interaction. The task is to detect and

*This research has received funding from the EU-FP7 STREP project EARS (#609465) and ERC Advanced Grant VHIA (#340113).

*ICMI* '15, November 09-13, 2015, Seattle, WA, USA
.

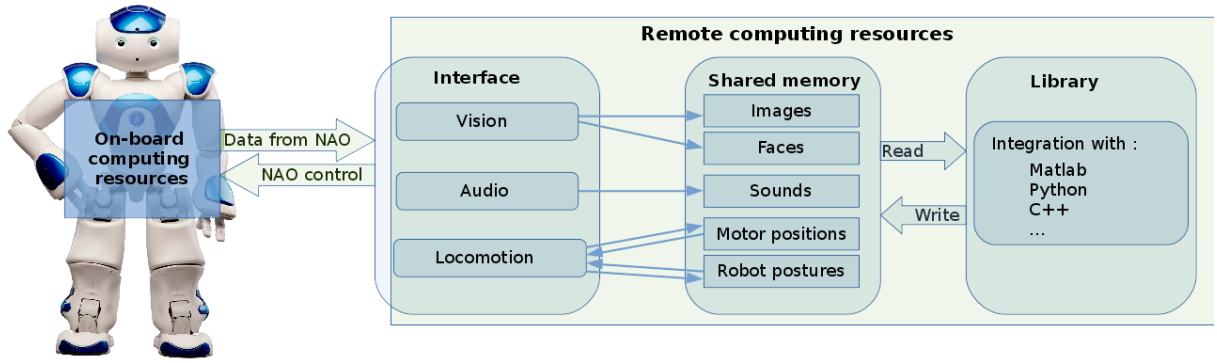



**Figure 1: Overview of the proposed distributed architecture that allows fast development of interactive applications using the humanoid robot NAO.**

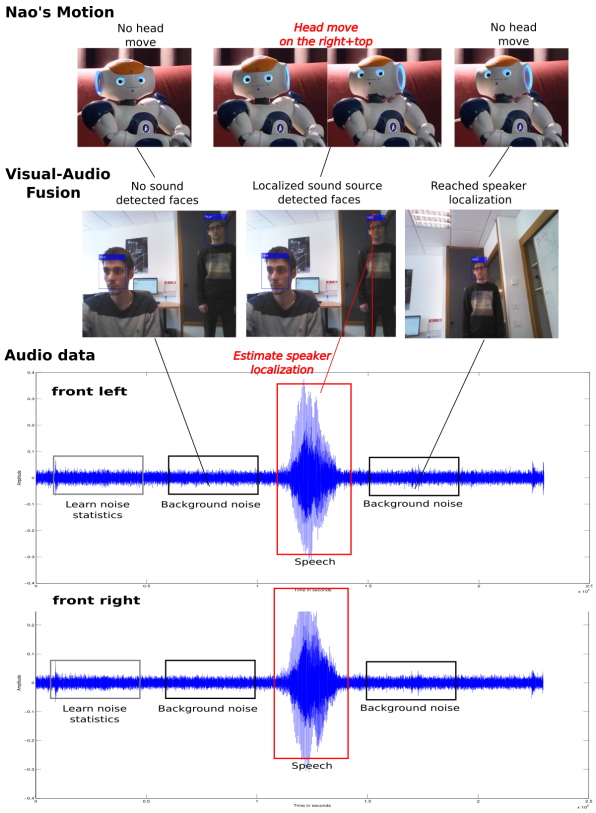



**Figure 2: Overview of the audio-visual localization algorithm that involves audio signal processing (voice activity detection, cross-correlation, estimation of time-difference of arrival followed by direction of arrival of the speech signal) computer vision (face detection and localization) and control of the robot's head motions.**

localize a sound source, to turn the robot's head towards the sound direction, to possibly detect a face in this direction and to raise the robot's head towards the perceived human face, Fig. 2. The overall integration is done in Matlab, which offers the possibility to develop sound-source localization method, which in turn uses Matlab's signal processing toolbox (voice activity detection, multi-channel cross-correlation, short-time Fourier transform, etc.). The Matlab computer vision toolbox is used for face detection. The control of the robot head is done as above using the robot-posture control module.

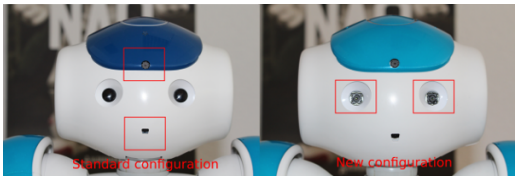



**Figure 3: The standard NAO head (left) includes two vertically aligned cameras with no common field of view (top camera is heading upwards while bottom camera is heading downwards). Recently, Aldebaran Robotics developed a prototype *stereo head* (right). The wide field of view shared by the cameras enables stereo matching and stereo reconstruction.**

- *Stereo reconstruction.* One important feature of HRI is enabling a robot to understand the spatial layout of the observed people. This allows a wide range of possibilities, including the recognition of human actions and gestures, configurations of groups of people, etc. For this purpose we use a stereo head recently developed by Aldebaran Robotics, Fig. 3. The on-board image acquisition module allows to transfer synchronized image pairs at 25 FPS at VGA resolution. The camera pair is precisely calibrated using any camera calibration toolbox currently available (computer vision Matlab toolbox, OpenCV, etc.). In this example we used OpenCV stereo matching and reconstruction modules. The visualization of the stereo reconstruction results are done in Matlab. A dense depth map is delivered at 10 FPS.

## Software Download

The proposed distributed architecture is compatible with the version 5 of NAO. C++ source modules are downloadable from `https://team.inria.fr/perception/en/naolab/`.