MULTIMODAL MALL ENTERTAINMENT ROBOT mummer-project.eu

Grant No. 688147. Project started 2016-03-01. Duration 48 months.

DELIVERABLE D2.1

AUDIO-VISUAL ALGORITHMS FOR PERSON TRACKING AND CHARACTERIZATION (BASELINE)

Jean-Marc Odobez (Idiap), Natalia Lyubova (SBRE),
Olivier Canévet (Idiap), Kenneth Funes Mora (Idiap),
Weipeng He (Idiap), Angel Martinez Gonzalez (Idiap),
Jean-Marc Montanier (SBRE), Marc Moreaux (SBRE)

Beneficiaries: **Idiap Research Institute** (lead), SoftBank Robotics Europe
Workpackage: **Active Multimodal Sensing and Perception**

Version: Draft
Nature: Report (R)
Dissemination level: Public (PU)
Pages:
2017-3-3

This project has received funding from the European Union's Horizon 2020 research and innovation programme.

Contents

1 Executive summary

Work package 2 (WP2) is about sensing people in general, that is, maintaining a representation of the persons around the Pepper robot, with a dedicated attention to people suceptible of interacting with Pepper, or those who are or have been interacting with it. This requires the design of several audio-visual algorithms to detect people, track them, re-identify them, and detecting their non-verbal behaviors and activities, and guessing about their position/behaviours even when they are not seen. At the same time, the representation of people needs to be defined and shared with other modules using it, or responsible of inferring other knowledge about people (typically in dialog for instance, to define a person's goal in the interaction for instance).

This deliverable presents the first year activities along these lines: getting acquainted with the Pepper robot and understanding its sensing capacities; defining the API to be used by partners to develop their own modules, e.g. interaction; and developing the first instances of this API using available AV-processing tools while investigating new algorithms and working towards new modules to be integrated in the future.

2 Introduction

In this section, we first remind the goals of the WP2 package, both in terms of software and scientific objectives, and then introduce the work done during the first year.

2.1 WP2 objectives

Main objectives. The main objective of WP2 is to provide Pepper with the multi-modal sensing and perception capabilities to understand the social scene and to allow it to pro-actively interact with people or groups of people in function of their physical accessibility, social availability, engagement action attempt, and in general monitor the interaction.

To this end, our aim is to design principled models and efficient algorithms for situated perception. We will develop a scene and person management module fusing the different streams of information (audio and RGB-Depth sensors, proximity, range sensors, robot state, or behaviours) to detect and track the perceived state and activity of the different people in the scene along several dimensions:

- physical states (location, trajectory, pose, speaking status),
- social category (gender, age group),
- non-verbal cues related to body language including attentional cues and basic gestures, actions, or emotions.

The semantic labels for the categories (social or gestures) will be decided based on the literature, or on results of the co-design process and the selected scenarios.

Research strategies. Broadly speaking, there are several research lines that we intend to explore during the project:

- **deep learning:** investigate the design and integration of computer vision methods combining deep learning and probabilistic methods.
- **context aware sensing:** investigate advanced probabilistic perception strategies leveraging on the interaction context as known by the robot (when and how he moves, gestures, speech, to whom he addresses, which objects he is referencing, etc.) to improve sensing and set priors on peoples behaviours (motion, speech, ...) and exploit them to improve interpretation and model adaptation.
- **active sensing and interactions:** uncertainties and reliabilities about the different cues will be provided to conduct a finer inference of higher social and interaction state allow the interaction engine to select the appropriate action. This will include actions that can improve sensing, like asking users to move closer, repeat, or use a different modality (e.g. text provided by touchscreen) if their voice cannot be reliably recognized due to a high level of noise, or overlapped speech or other challenges exposed by environment.

Note that these research strategies (and their specific instantiations) may evolve depending on the evolution of the research field.

2.2 Achievements

During the first year, we have worked on several directions towards the above goals. We summarize them below, and provide pointers to the rest of the documents where more details can be found.

- **Perception interface (API).** One of the main WP2 goal is to provide the sensing outputs to the partners working in WP3, WP4 and WP5. We have thus worked on this issue during the second part of the year to ensure a good integration of the perception modules and their easy updates during the course

of the project. The approach and first instantiation based on NAOqi perception algorithm is presented in Section 3.

- **Depth sensing and recording.** During the first year we have evaluated the sensors from the robot and their compatibility with the intended and planned algorithms. Two lines of work have consumed some resources, mainly evaluating the depth sensing quality, and the audio-visual capacities to record data for learning algorithms and benchmarking purposes. They are summarized in Section 4.
- **Perception: tracking methodology overview.** Besides the API developments, we have been working on the development and/or the integration of several perception components useful for the project. The rest of the deliverable summarizes them. In Section 5, we provide a brief description of the perception modules currently used by the system and based on NAOqi, as well as the ongoing developments and plans for the second version of the multi-person tracking algorithm in the project.
- **Perception: Close/medium range sensing.** On the side of a multi-person tracking module required to localize and keep the ID of all people in the scene (at least those who are close to the robot), we are working on algorithms and methods that can perform a more accurate analysis of people and of their non-verbal behaviors. In particular, we have worked on improving our 3D head pose and RGB-D gaze estimation modules, and preparing a first real-time demonstration ready for integration with the Pepper robot. This line of work is described in Section 6
- **Perception: Human activity recognition.** Recognition of human activities is useful for a better understanding of people near the robot. We have started to work on human activity recognition based on data from sensors embedded on Pepper and our approach is described in Section 7.
- **Perception: 3D skeleton tracking.** Getting a precise skeleton of people and on-board of the robot in real-time is important for most perception task. This year we have worked on this topic and the methods and results are presented in Section 8.
- **Perception: Audio sensing.** While external resources (software) are planned to handle audio processing and particularly ASR, the inference of sound localisation and of the speaking status of people in multi-party interactions is still a rather important and unsolved problem in unconstrained environments. We are currently working on this issue, and a sketch of our approach and current status is provided in Section 9.
- **Conclusion and future work.** Finally, Section 10 will provide a brief conclusion of the work as will list the plans in both research and developments for the next period.

Overall, the work in the work package has progressed according to plans, with a first system available, and other modules ready to be integrated. No major deviations have been observed.

3 Software: Interface specification, instantiation, and processing strategy

The perception interface aims at (i) defining the information and perceptual processing elements which are relevant to the interaction modules. Note that this includes as well the information that the perception module need to receive in order to achieve its task. (ii) providing the detection and analysis results obtained from processing the videos and audio streams in a format that will be used by these modules, like social state analasis, or action and motion planers.

Thus, in the following, we first present in section 3.1 the elements used to specify the current API. In a second step (Section 3.2), we briefly summarize the method used by Idiap to implement a first version of the defined API, based on the ROS-Bridge layer of SBRE and the NAOqi perception modules. Finally, as perception is often a time consuming task, we present in Section 3.3 the envisioned strategy to stream and process the data, in order to allow the design of state-of-the-art algorithms while fulfilling the deployment constraints.

3.1 API overall specification

3.1.1 Introduction

Perception requires two different modalities to analyse the environment of the robot and provide an interpretation of people in the world. The first one is the sensor outputs, while the second one is the interaction context (see input of "perception module" of figure 1):

- **Sensors.** These comprises the main video, depth, and audio streams which are processed either directly on the robot, or on an external computer after streaming through WiFi or cable. In addition, the robot states (for proprioception) and potentially other sensor modalities (like sonars) will be used based on needs.
- **Interaction context.** As perception is not only a bottom up process, in addition to the sensor data, it is important to get extra information regarding the task context or the state of the interaction. This will play a role for analyzing the situation and making perception aware of different factors which can influence what is done, e.g. focusing the computational resources on specific perceptual aspects. In particular, we are interested in the following context:
  - Robot motion: we need to know which motion/gesture is planned or currently performed. In particular head motion/gestures need to be known, as they affect a lot the sensing.
  - List of people Pepper is currently interacting with. This is important to balance computational resources (on which person should finer analysis be conducted), or to understand which person should be kept in memory for further re-identification for instance.
  - Attention analysis: to achieve this task, the module need to know which are the target to be considered. This could be provided as a list of targets, either defined through their 3D location, or as a set of identifier (person identifier, or object identifier, if such a list is created somewhere).
  - Robot speech and dialog act: this is the same in the audio domain, we want to know if speech (and in general audio) is planned or is going on, since this will affect the audio perception, and can be used as prior for interpretation.

During year 1, we have left aside the integration context input and worked mainly on specifying the module output. We thus present this part in the next subsection.

3.1.2 API specification

The main idea for the specification has been to employ a modular approach (i) publish first and continously information regarding the physical states of all detected and tracked persons, where we aim at maintaining the

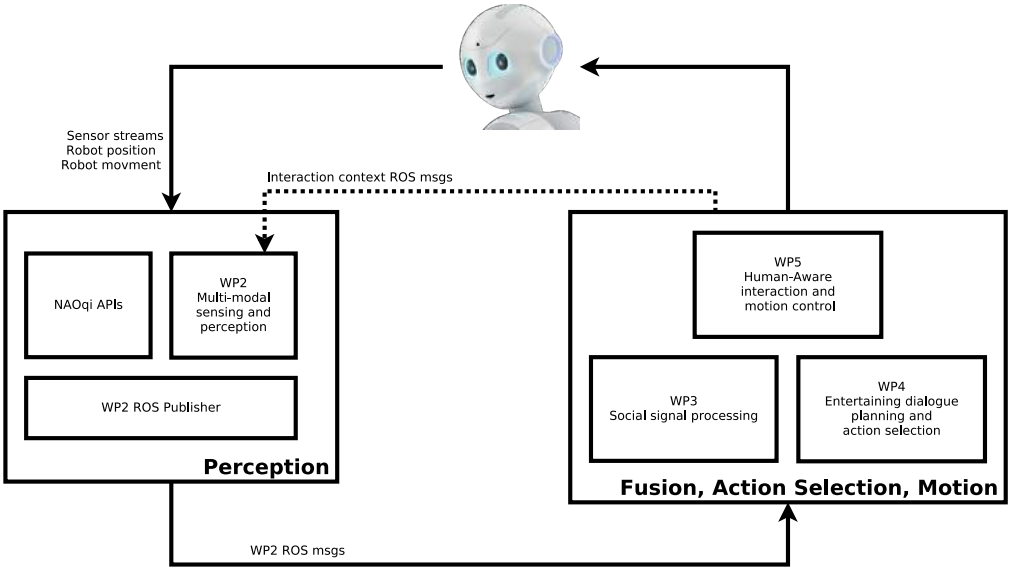


Figure 1: API and interaction with other main processing tasks. WP2 receives the streams to process (audio, video, depth, egomotion) and contextual information from other WPs, and produces a stream of representation of people in the environment.

same ID for each person; (ii) on top of this, process and publish additional information related to people and which might be the results of different analysis modules: Attention (linked to head pose and gaze analysis, and requires potential input about person/objects other than pepper and its screen); Head gestures (e.g. nod); Face analysis (gender and expressions); Speaking activities (who is speaking, informing about speaking turns). This is further detailed below.

Person tracking. Tracking requires to detect people (bodies, faces, sounds), and associate them over time. A main difficulty in keeping a single person ID is (1) some latency, i.e. in general we may require several detections before deciding that we have indeed a person, i.e. a detection is not a false alarm (2) when people are lost for some time (e.g. they are out of the field of view or occluded), whenever we detect him/her again and create a new track or person ID. It may then take some time before we re-identify the person as one already seen. We will have thus to handle such situations (i.e. two persons ID corresponding to the same person). Several mechanisms will be used to take care of this point.

Attention, Gaze. The main goal will be to provide information regarding the gaze and attention of people. The list of tracks/person for which such information is available should be listed. Regarding attention, for each identified 3D target of interest, we aim at providing the probability that the person is looking at it. This means that when targets are close in space, they can all have a relatively high probability of being looked at (or not). Accuracy (and uncertainty) will depend on available information (head pose from vision only, head pose from 3D, gaze using eye information), amongst others.

By default, if the gaze analysis is done for a person, its probability of looking at Pepper's head or Pepper's screen will always be provided. For other targets, their IDs will be used to identify them, provided their location has been given as input context.

Face analysis and expression. This relates to the social attributes (age, gender) and expressions (with an emphasis on smile) of each person.

Voice activity. The main goal is to provide information regarding the speaking status of people, analyzing the turns, and in subsequent phases, identify further communication attributes such as the potential addressee of the speech for instance.

3.2 First baseline API instantiation based on Pepper perception modules

3.2.1 Implementation

Through collaboration with partners regarding the kind of information required to make the robot interact with the people, we have designed an interface of ROS messages (see appendix A). Among the outputs required for interaction, we can name the position of people, whether people are looking at the robot, if they were identified in the past, or their facial expressions. These informations are gathered in thematic messages (gaze, expression, track, sound) so that the action planners simply subscribe to the message of interest.

The most important benefit of such an interface is that as the perception algorithms are evolving, the interface remains the same, and the modules built upon the perception module remain unchanged, although some additional elements could be added in the future depending on the needs of the other modules.

The current implementation of the API fully relies on the NAOqi components already available. The following data provided by perceptual modules in NAOqi can be accessed via the provided ROS bridge (see section 3.2.2):

- an array of **detected people** with the following data for each person:
  - person position, orientation, distance to the robot,
  - if face detected
  - gaze angle, head angle, a score of looking at robot,
  - age, gender, smile degree and their confidence,
  - other expression properties: neutral, happy, surprised, angry, sad;
- an array of **detected faces** with the following data for each person:
  - face recognition score, along with face shape and size;
  - positions of facial characteristic points for both eyes and eyebrows, nose, and mouth.

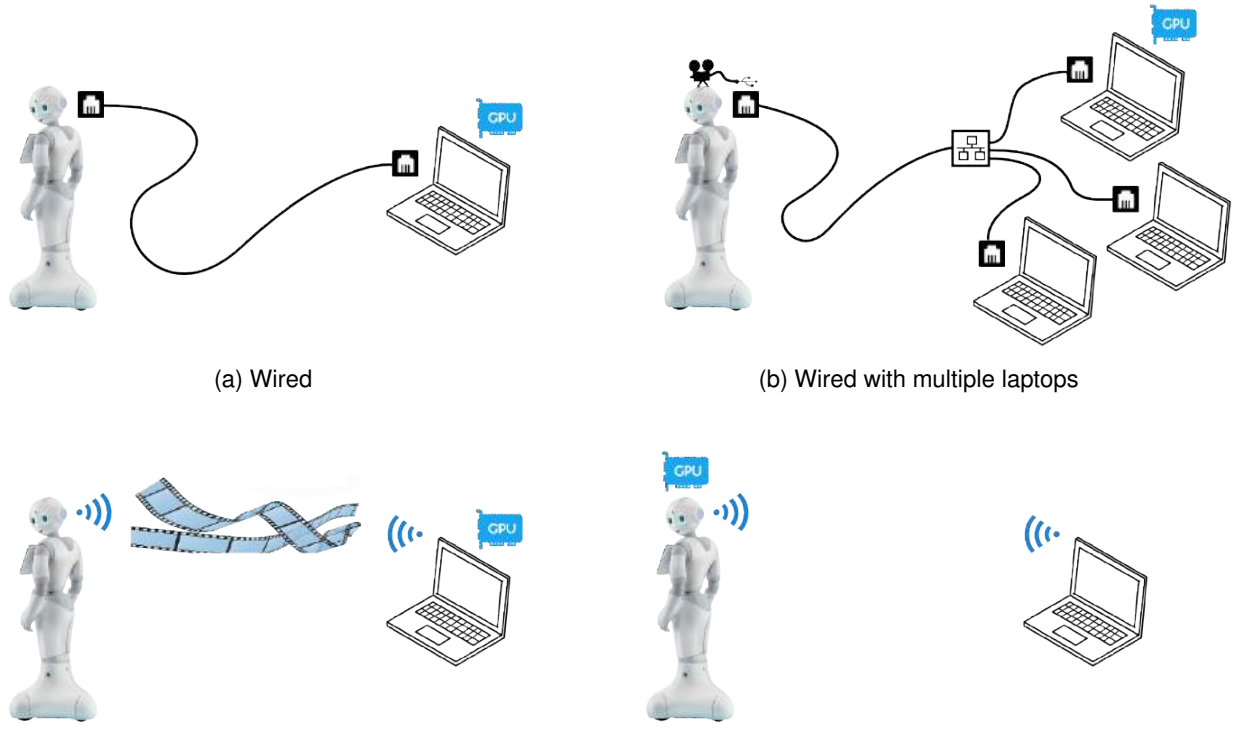
3.2.2 Ros-Bridge

In this project, SBRE provides initial components for people perception that are integrated and delivered in NAOqi running on a robot. Among perceptual components, there are detection of people, faces, gaze analysis and analysis of facial characteristics and expressions that are described in Section 5.1. All these perceptual components are running locally on the robot and take their inputs from sensors embedded on Pepper. Both the outcome data from the perceptual components and the raw data from sensors embedded on Pepper can be accessed via NAOqi API[1]. In addition to NAOqi, SBRE provides interfaces to access the most useful data and functions from NAOqi via a ROS environment (on demand of partners) using one of the following ROS Bridge packages:

- *Naoqi Driver*[2]: a generic ROS bridge that is common for all SBRE robots such as Pepper, Nao or Romeo;
- *Pepper bringup*[3]: Pepper-specific package that is based on NAOqi Driver and provides additional functionalities for Pepper robot, such as perception functions including data registration for depth and RGB embedded sensors.

The goal of this ROS bridge is to provide an access to the data from the robot (its sensors and motors) and from NAOqi as efficiently as possible. Thus, ROS bridge gets the data straight from the lowest levels of NAOqi hence ensuring low latency and low CPU usage. It publishes all available sensors data to ROS as well as the robot's position trying to be as close as possible to ROS standards by exposing standard topics. The ROS Bridge used in this project is initially an open-source project adapted for the needs of this project to ensure that partners can access all required data from the robot and its software as efficient as possible and can control the robot via ROS.

[1] http://doc.aldebaran.com/2-5/index_dev_guide.html
[2] https://github.com/ros-naoqi/naoqi_driver
[3] https://github.com/ros-naoqi/pepper_bringup



(a) Wired

(b) Wired with multiple laptops

(c) WiFi connection

(d) WiFi with embedded GPU

Figure 2: Perception processing strategies planned in the project. (a) The default will consists of using a wired connection from Pepper to the processing laptop with GPU. (b) Potentially, the wiring may by shared by multiple laptops (e.g. to share with other WP processings). (c) If the WiFi connection and bandwidth allows the streaming of images and depth and in general data at a sufficient frame rate, this option will be used. (d) Ideally, if an embedded GPU is available, the processing can be done all on-board, or with some WiFi connection to the laptop, but the video is not transmitted, only meta-representations.

3.3 Computation strategy

This section describes the setup of computer hardware and software required to achieve the best perception capabilities for the robot. Figure 2 shows the various configurations which can be considered. The current system used for development is depicted in figure 2a: the robot is plugged to the computer and the video and audio streams are acquired through the Ethernet cable. The current WiFi system does not allow a sufficient frame rate for real time application, hence the need for a cable.

An important part of the perception module relies on computer vision. All state-of-the-art algorithms require a large amount to computational power, which is achieved with graphics processing unit (GPU). Among widely used methods, we can name AlexNet for image classification [13], residual networks [11], or the convolutional pose machine [4] for human pose estimation. A GPU is therefore needed to achieve the best results on the perception part. The current experiments rely on a GPU hosted by a remote laptop (figures 2a and 2b), but we could imagine a on-board GPU (figure 2d) to free ourselves from an external laptop for the computation. In this case, the laptop could be use for log checks.

In the current setup, the Ethernet cable enables limited movement of the robot in a close area around its position. The WiFi alternative would be preferable to the wired one, to enable greater mobility, but requires the streaming of compressed images which is not the case at the current state of the project (see tables 1 and 2 for the streaming rates with and without compression).

3.4 Conclusion

The output of the perception module is broadcast to the other modules (dialogue, motion planning, interaction control) through ROS messages (see figure 1). The other modules subscribe to the corresponding topics and take action depending on the content of the messages. The current interface has been tested and validated by the other partners, who are currently using it in their implementation. The next chapters are dedicated to detail the algorithms used for perception, which actually fill the ROS messages of this interface.

4 Pepper depth sensing and AV recording capacities

Depth is an important cue that provides essential information for 3D scene understanding and allows fast processing. Noticing that the quality of depth available with Pepper was rather low compare to the normal ASUS Xtion sensor it was built upon, Idiap has spent efforts on investigating this issue, collaborating with and providing feedback and inputs to SBRE. In February 2017, SBRE realized that the QVGA data provided by Pepper actually corresponded to an upscaled depth map of a QQVGA (Quarter of QVGA) depth map, produced form the sensor in order to minimise data transfers. The data gathered without this compression seems to be of a better quality and should be useful now for processing algorithms.

In addition, for data collection and benchmarking purposes, recording capacities (here streaming and data synchronization) of all data streams is very important. Experiments along these lines are summarized in Section 9.2. While streaming capacity should be enough (especially with the inclusion of data compression before streaming as planned in WP6), the synchronization between the RGB and Depth streams may remains an issue for joint processing of the two streams.

4.1 Depth sensing quality

4.1.1 Introduction

Advantage of depth sensing. Depth sensors make direct measurements of the environment's 3D geometry. This information is strongly complementary to the standard vision modality and has been exploited more and more in recent years to address a wide variety of computer vision problems, such as head pose estimation, body detection and pose estimation, object recognition, etc. This has also been facilitated through the availability of cheaper and acceptable quality sensors, such as the Microsoft Kinect (1 and 2), Asus Xtion, Primesense, Intel RealSense, etc. Moreover, when based on active technologies like structure light or time-of-flight, these sensors are fairly robust to the object's texture and scene illumination, as opposed to stereo setups.

Depth usage in WP2. The depth sensor embedded in Pepper's head is therefore of high value to address the tasks defined in this work package. The objectives are diverse:

- we aim to employ methodologies developed at Idiap, for head pose and gaze estimation, which employ RGB and Depth modalities jointly in order to recognise people's non-verbal cues and attention with higher granularity when it comes to a close interaction with Pepper;
- depth can help to accurately estimate the position of elements in the 3D space, this is of value for 3D body pose estimation. Note that methodologies like convolutional pose machines (see Section 5), although highly promising, are capable of detecting joints in the 2D image coordinates domain, but it is not trivial to retrieve a plausible 3D pose explaining the 2D configuration, considering aspects such as scale and person specific body geometry;
- the fast and onboarding of people can be done faster, through robust detection in the depth domain;
- the additional information provided by depth can help to reduce the processing time, and therefore make a more natural interaction with faster robot response.

The depth sensor on Pepper, according to the specifications, is an embedded Asus Xtion camera. In the process of employing this data for developing WP2 tasks, the data quality was found to be of significantly lower quality than an off-the-shelf Asus Xtion and similar consumer depth sensors. In this section, we describe the steps that were taken to validate these issues and to find possible solutions.

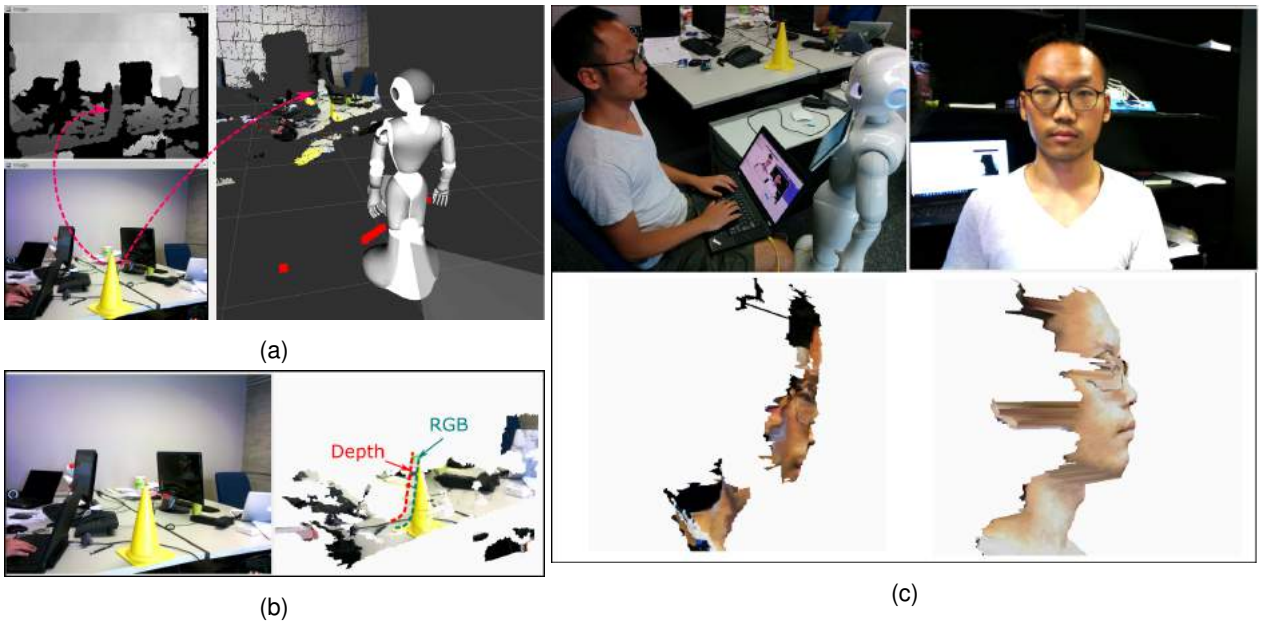


Figure 3: Qualitative analysis of the RGB-D camera provided by the Pepper robot at the start of the project. Figure (a) shows the colored point-cloud rendered by ROS, note how the cone does not get the yellow texture mapped into it when 3D rendered. Figure (b) shows a textured 3D mesh rendered from the RGB and Depth data using Pepper's camera calibration parameters using the rgbd module for visualisation. Note the misalignment between shape (depth) and texture (RGB) due to inaccurate calibration parameters. Figure (c) Profile view of a 3D rendered face using Pepper's sensor (bottom-left) and an Asus Xtion (bottom-right) taken for the same person at a similar distance. While misalignment issues of RGB and depth hold (but were reduced at this stage), note how the Pepper's sensed point cloud does not convey the facial shape, as opposed to the data captured with a commercial Asus Xtion, where the facial profile shape is well observed (nose shape, mouth, etc.).

4.1.2 Pepper's depth sensor assessment and improvements

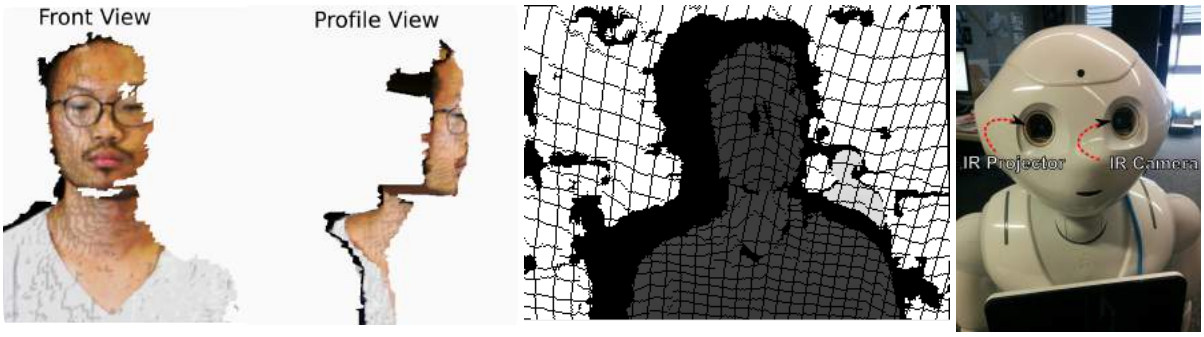
The quality of a depth sensor, intended to be used jointly with an RGB sensor, is evaluated according to following criteria: the depth noise (along the camera optical axis), the depth tangential and radial distortion in the image, the calibration with respect to the RGB sensor as well as the resolution of the depth data, linked to the density of the associated point cloud. In this section we address these elements in the order they were addressed.

Initial evaluations and RGB-D calibration. For the initial collection of data we used NAOqi 2.4, as provided with the Pepper robot shipped to Idiap. We used calibration parameters set through the specifications for Pepper and ROS definitions, as recommended by SBRE. This data was then processed to visualise the equivalent textured-binded 3D meshes of the scene. The following tools were used for the rendering:

- Robot Operating System (ROS). We used the modules for visualisation of point clouds, with the associated color from depth-to-RGB mapping. This tool gathers data directly from NAOqi.
- *rgbd* module[4]. We used Idiap's in-house built tools. This module form the basis for some of our algorithms.

In Figure 3 we show representative results of the initial data quality. Figures 3a and 3b indicates the effect of missing appropriate calibration parameters. We used both visualisation tools to discard software usage related errors, and the results were consistent. SBRE then provided instructions on how to calibrate the RGB-D ensemble from Pepper. In short, the process consisted on a standard stereo camera calibration procedure using a checkerboard pattern, while the IR projector in Pepper is covered to avoid capturing the scene with the structured illumination pattern. Therefore, it is a calibration between the IR camera and the color camera. We obtained noticeable improvements in terms of the color and depth alignment (through intrinsics and extrinsics parameters),

[4] https://www.idiap.ch/scientific-research/resources/rgbd-a-python-based-rgb-d-data-processing-module



(a) (b) (c)

Figure 4: Diverse steps that were taken towards improving the depth sensor in Pepper: (a) RGB-D calibration; (b) Depth map registration; (c) Removal of lenses from the depth sensor composed of an IR projector and an IR camera.

(a) (b) (c)

Figure 5: Data rendering and typical head pose tracking results for the same subject using three devices. The 3D rendering images correspond to a similar distance across the three devices (0.8 meters), where as the head pose tracking results correspond to sensing at a distance of 1.2meters. (a) Kinect 2 (for Windows); (b) Primesense Carmine 1.09 at QVGA depth resolution (320x240); (c) Pepper, at QVGA depth resolution (320x240).

but the calibration parameters have no influence on the depth quality itself.

We focused on analysis the potential of the data quality for tasks such as head pose estimation. Fig.3c indicates why the depth sensor was providing a poor quality for tasks related to face analysis. The face, when sensed through Pepper, is observed as a blob lacking shape information of important facial features like the nose, or mouth shape, as opposed to the equivalent off-the-shelf commercial device (Asus Xtion). The same holds after calibration, as seen in Figure 4a, which corrects alignments between color and depth, but the sensed facial shape still lacks facial features needed for face analysis from depth.

We also compared differences between using the color spaces specified in the NAOqi SDK AL::kRawDepthColorSpace and AL::kDepthColorSpace, but we did not observe differences.

QVGA, VGA, and registered depth. The default depth resolution for the sensor in Pepper is QVGA (320x240). A hypothesis from SBRE was that higher resolution data (VGA, 640x480) was needed. This is an option available with Asus Xtion sensors, but not available by default at Pepper's device through NAOqi. Therefore, the Pepper robot at Idiap had to be upgraded into NAOqi 2.5 drivers, provided by SBRE. Nevertheless, we found that although some improvements were observed, these were not enough to be used with the algorithms for WP2 (in particular, head and eye gaze tracking).

Registered data did not improve the data quality either. The registration process consists on projecting the depth map into the RGB camera such that there is a one to one pixel alignment. However, this does not affect the depth noise along the camera optical axis, thus it did not improve the data quality. Moreover, as seen in Figure 4b, a regular pattern of missing depth pixels appear due to sampling issues.

Lens distortion. In following steps we found that planar objects (like walls) were highly curved when 3D ren-

Figure 6: Data example collected at SBRE when using true QVGA, rather than upscaled QQVGA data.

dered using a live stream from Pepper's depth sensor. Therefore, SBRE agreed to remove the lenses from the Pepper robot at Idiap, as shown in Figure 4c, while in parallel searching for other materials that could maintain Pepper's look for the end customer while removing optical distortions on depth sensing. A clear improvement was observed for the 3D data in general as planar objects were indeed planar in their 3D rendered version. However, the fine depth sensing needed, for example, for face analysis, did not improve and remained similar to Fig. 4a.

Comparison to other consumer sensors. We made recordings with other off-the-shelf consumer RGB-D sensors in order to establish a baseline for the depth data quality when it comes to face analysis. This data was provided to SBRE for reference. In addition, SBRE provided to Idiap data recorded with their Pepper in order to analyze whether the problem was in the device itself.

Samples of these data can be seen in Figure 5 as well as a typical results for our head pose tracking algorithm. Note that the Kinect 2 as well as the Primesense Carmine 1.09 device maintain a consistent, and sharp estimation of the facial shape. The Primesense in particular is comparable in quality to an Asus Xtion device. The results on Pepper, consistent with our previous results, provide large facial distortions. The influence on the head pose tracking is noticeable, as the head pose tracking algorithm fails (cf. Fig 5c) due to the lack of geometric face support, suggesting the strong need for higher quality depth data.

Effective QVGA depth resolution. In a recent update SBRE has concluded that the provided QVGA data provided by Pepper actually correspond to an upscaled depth map of the QQVGA depth map, done in order to minimise data transfers. SBRE provided data to Idiap in order to make evaluations after making this correction and capturing true QVGA data. Qualitative results can be seen in Figure 6, suggesting a significant improvement on the depth sensing quality. Further tests are yet pending using Idiap's Pepper robot.

4.2 Streaming capacities and synchronization accuracy

Introduction Recording is a basic and essential part of perception research. Only with enough high quality recorded data, we are able to train models and precisely evaluate them.

In preparation for future data collection, we conducted several tests to understand the capabilities and limits of data streaming and recording of Pepper. More specifically, we evaluated the following capabilities:

- Audio-visual synchronization,
- Data streaming rate under different conditions,
- Synchronization between RGB and depth cameras.

We used *rosbag*, the standard recording tool for Robot Operating System (ROS), to record samples for evaluation. *rosbag* saves data in original format together with the their time stamp. And the recorded data can be played back as it was recorded.

In addition to evaluating the streaming and recording capabilities, we have also developed several tools related to recording. These include tools for audio/video data conversion, visualization and audio playback.

Audio-visual synchronization. We tested the audio-visual synchronization by recording a 5-minute video of a person speaking to the robot, and the synchronization is evaluated by subjectively watching the lip movements and listening to the audio recording simultaneously. As the result, we found that there is no noticeable delay between audio and video data.

Data streaming rate. We tested the data streaming rate for recording via *naoqi_driver* (ROS interface of NAOqi) under different conditions. The conditions include different connection settings: via a Gigabit Ethernet cable, 100Mb/s Ethernet cable or via WiFi (802.11n standard); and different robot states: with autonomous life on or off. The following data are streamed simultaneously:

- Front RGB camera images and camera info.
- Depth camera images and camera info.
- 4-channel audio from the microphone array.
- Robot joint states and ROS frame transforms (topic '/tf').
- Sonar, laser and infra-red data from robot base.

The streaming rates for videos are shown as in Table 1. Note that in the current NAOqi system, the video data is not compressed before streaming.

Connection	Gigabit Ethernet		100Mb/s Ethernet		WiFi	
Autonomous Life	off	on	off	on	off	on
QVGA RGB + QVGA DEPTH	30	18	18	12	4	1
VGA RGB + QVGA DEPTH	30	15	8	8	1	$<$1

Table 1: Video streaming rate without compression. The numbers in the table indicate frames per second (fps). Note that as mentioned in Section 4.1.2, the QVGA depth images are actually upsampled from QQVGA depth images captured by Pepper.

The audio streaming rate is fixed under any condition at 4-channel 48000Hz without data loss. And the other data are streamed without interfering the video and audio streaming.

In addition to streaming with *naoqi_driver*, we also tried other methods. One of which is running a GStreamer video server on the robot and the server streams RGB video data compressed by JPEG. This approach significantly improves the streaming rate. The result is shown in Table 2.

Data streamed	Ethernet	WiFi
QVGA RGB Compressed	30	24
VGA RGB Compressed	30	15

Table 2: Video streaming rate with compression. The numbers in the table indicate the frames per second (fps).

The results shows the video streaming of current NAOqi system is not fast enough for most of the tracking applications. Before the systems is improved by either using compression for streaming or upgrade WiFi hardware, we can only work with the Ethernet cable plugged to the robot.

RGB-D synchronization issue. We measured the delay between the arrival time of RGB and depth images, and found that it varies from 10 to 20 milliseconds. While this might be sufficiently good for some late RGB and depth fusion modules (eg extracting depth measures from localized body parts, to estimate people 3D pose), it will be insufficiency for combined depth and RGB processing, especially gaze tracking which requires very fine synchronization between RGB-D (since in 10 or 20 milliseconds, people will move slightly, and an offset of 1 or 2 pixels in eye localizatin means 5 to 10 degree error).

5 Perception and tracking methodology overview

One main objective is to localize people and track them robustly (tracking ID switching is a recurrent problem in multi-party HRI scenarios) and then extract person related attributes, like non-verbal behaviors, emotions, activities, etc. The two stages might be fairly independent. In particular, for computational issues, the second one might be done with high precision only with people that matter for the interaction (c.f. API specification). In that view, the current approach in the project is thus to:

1. design a multi-person tracking approach incorporating a generic processing of attributes;
2. design dedicated or more precise person-specific processing modules for attribute recognition which can be applied to a shorter list of people of interest. This is the case for instance of the gaze and attention modules (Section 6), for the recognition of activities (or for the ASR part, using external softwares).

In this section, we present methods which relate more to the first category. In particular, we briefly describe those which have been used in our baseline system (c.f. first API instantiation as described in Section 3.2) and are based on the perception modules available on Pepper (Section 5.1). In addition, we also present in Section 5.2 the tracker currently under development and which should be available in March or April to partners.

5.1 Pepper perception modules used with baseline API implementation

In this project, SBRE provides initial components for people perception integrated and delivered in NAOqi running on a robot. Among these components are people detection, face detection, gaze analysis and analysis of facial characteristics including facial expressions, that are described in following subsections. All these components are based on data from sensors embedded on Pepper and all outcome data can be accessed though NAOqi or ROS Bridge packages described in Section 3.

People perception in NAOqi controlled by PeoplePerception module that keeps track of people detected near the robot and provides basic information about them through NAOqi ALMemory keys and raised events; the whole list of memory keys and events is provided in NAOqi documentation[5].

5.1.1 Methodology

In the **PeoplePerception** NAOqi module, potential humans near the robot are detected based on visual cues and data acquired from RGB cameras and 3D sensor embedded on Pepper. People detection begins with detecting a human face from RGB data. Once a face is detected, it is associated with a 3D pointcloud and tracked over time. Thus, Pepper is able to detect a person only after seeing his/her face. However once the face is detected, Pepper can keep tracking the person even without seeing his/her face. All people detected by Pepper are stored in the people population. Newly detected people are associated with previously known people if possible. If a new person cannot be associated to a previously known person, then he/she is added to the population. If a tracked person is visible then he/she is stored in the visible people list; when the tracked person gets out of the field of view of Pepper then he/she is moved to non-visible people list but not immediately removed form the population, since the person can disappear temporally as a result of the robot's movements. The person is removed from the population, only if he/she has not been seen for a while.

The **FaceDetection** NAOqi module is responsible for detecting human faces and optionally recognizing faces if in front of the robot and if possible. The module is based on a third-party library for face detection/recognition provided by OMRON [14]. For each detected face, the module provides its position and a list of angular coordinates for faces features such as eyes, nose, and mouths. In addition to face detection, it is also possible to

[5] http://doc.aldebaran.com/2-5/naoqi/peopleperception/alpeopleperception.html

recognize faces, but it requires a learning stage. After the learning stage, the module returns a list of recognized people for every image. For further details, see NAOqi documentation[6].

The **GazeAnalysis** NAOqi module performs analysis of gaze direction for detected faces; it detects whether the persons eyes are open or close and estimates a confidence score describing the fact that the person is looking at the robot or not.

The **FaceCharacteristics** NAOqi module performs analysis of facial characteristic points and provides additional information about a person such as an estimation of age, gender, smile degree, and facial expression, if possible. Each estimated characteristic is coupled with a confidence score between 0 and 1. The age estimation is performed on each single image and not a sequence of images, thus the obtained result can change over consecutive images, and it is recommended to compute an age category over images rather than using our raw estimation. Among facial expressions, we distinguish neutral, happy, surprised, angry and sad expressions.

5.1.2 Performance and limitations

The performance of people perception components are influenced mainly by the robot's hardware limitations (i.e. limited processing resources that are shared between all software components simultaneously running on the robot) and also environment conditions. The following conditions makes people perception difficult:

- extremely strong back light, especially when the cameras are directly hit by sunlight,
- a human standing too close to a wall, an object of a human-size, or another person.

Face detection is working in the following conditions:

- face size: minimum requires face width is 20 pixels in the image; for an adult, it corresponds to a maximal distance about 2 meters when processing RGB images in QVGA resolution and 4 meters in VGA resolution,
- face orientation: works for +/-15 deg, if 0 deg corresponds to a face facing the robot's camera,
- face rotation in the image plane: works for maximum +/- 45 deg,
- lightning conditions: tested for 100-500lux.

The performance of facial analysis is limited in the following cases:

- a person wears accessories such as glasses or a hat,
- face deformation such as a pronounced grin or grimace,
- for young children with less distinctive facial features, the gender estimation is rather difficult.

5.2 Second API instantiation: multi-person tracking with convolutional pose machines (under developement)

In view of the above limitations, Idiap is working on a second version of the API. It relies on a previous multi-person tracking and analysis methodology, whose main capacities and processing components are described in Section 5.2.1, and which has been adapted and integrated within the MuMMER/Pepper platform. However, several improvements need to be made to handle the MuMMER scenarios: better detection robustness, and better re-identification.

In the following subsections, we thus present our approach to handle the robustness issue. To this end, we have decided to explore the use of very recent deep-neural networks (Convolution Pose Machines [4]) which have been shown to provide robust and accurate person detection. In Section 5.2.2, we first summarises the

[6] http://doc.aldebaran.com/2-5/naoqi/peopleperception/alfacedetection.html

main aspects of these networks. Then, in Section 5.2.3, we describe our work to improve the processing speed, latency, and exploit the CPM output in our tracker, as well as how to combine it with our previous tracker, which includes ways to obtain faster processing speed, and fusion with NAOqi modules for gender, age, and facial expression recognition.

5.2.1 Multi-person head pose and gesture tracking

For the new version, we will use as baseline tracker the head tracker developed in [12]. This tracker is based on texture and colour cues (skin) and efficiently combines the output of a face detector (OpenCV) with appearance cues. Tracking is formulated within a multi-person Bayesian filtering framework solved through sampling methods, also known as particle filter. Its main characteristics are the following:

- the state-space comprises both the 3D location (and speed) and head orientation, which are thus estimated jointly during inference, since both are strongly correlated (head appearance depend on pose, and vice-versa) [3].
- the likelihood thus integrates pose-dependent texture models and skin maps models learned from the POINTOING dataset [17]. The skin color distribution is learned from an automatic inference of skin, hair, cloth, and background color distribution using probabilistic index maps [18].
- dynamics and particle sampling: the system uses a switching dynamical model, drawing particles from several proposal functions: in addition to standard state-based prediction model (constant location speed model), it uses visual motion (estimated using a based on robust multi-resolution estimation framework) to handle dynamic motions (e.g. due to the robot head gestures) and face detections (both frontal and profile detections) associated to current tracks are also used [16].
- track management (initialization and termination). The OpenCV frontal face detector is mainly used to initialize track. A dedicated validation process is used to transform detections over time into validated tracks (to better remove false alarms), and decide about the re-association with existing person IDS. Similarly, a track termination model integrating multiple cues (dynamics, face detections, likelihood etc.) is used to remove tracks most probably corresponding to tracking failures [6].
- re-identification: a simple color model is used to re-associate tracks when they leave the field of view, and upon the re-detection of new persons.
- the tracker runs in real time (10 fps) which makes it efficient to track people for robot applications.

Figure 7 depicts an example of the output of the tracker, on which the tracked heads are displayed. In this image, given some input targets (Nao, persons, etc.), the tracker estimate where each person is looking at. On the image, the left person (ID 2) is looking at the robot (Nao), while the right person (ID 3) is looking at person 2.

5.2.2 Convolutional Pose Machines: person detection and body part localization

Convolutional pose machines (CPM) is a framework for the task of articulated multi-person pose estimation [20, 4]. Built upon the framework of pose machines [19], CPM consist of a sequence of convolutional neural networks that sequentially produce and refine 2D belief maps, on the image, for the location of human body parts and for the position and orientation of limbs connecting the body parts.

Figure 8 shows the overall architecture for the single-person body part detector network [20]. The network is composed of T stages. At each stage t, a full convolutional network $g_t(\cdot)$ produces a set of belief maps $\{b_t^p(Y_p = z)\}$ for each body part p representing the confidence that p is located at pixel $z \in \mathbb{R}^{w \times h}$. Each convolutional network $g_t(\cdot)$ uses image features and contextual information from previous stages to produce and

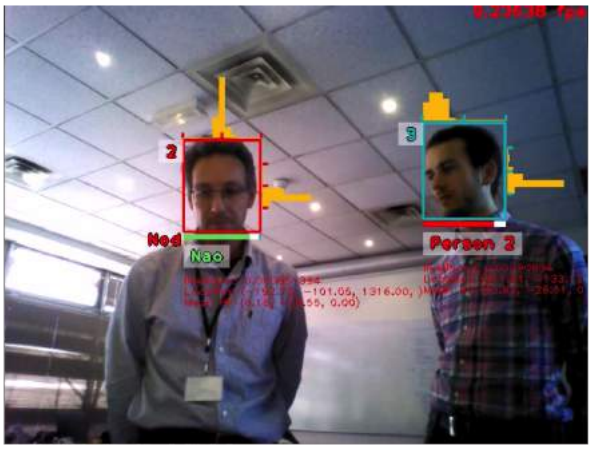


Figure 7: Output of the Idiap tracking algorithm. Tracked persons are depicted with a square around the head; the numbers are the track ID of the persons. The text below the tracks indicates which target one person is looking at. Head pose (vertical and horizontal distributions, yellow bars) and gesture recognition (Nodding) are also computed.

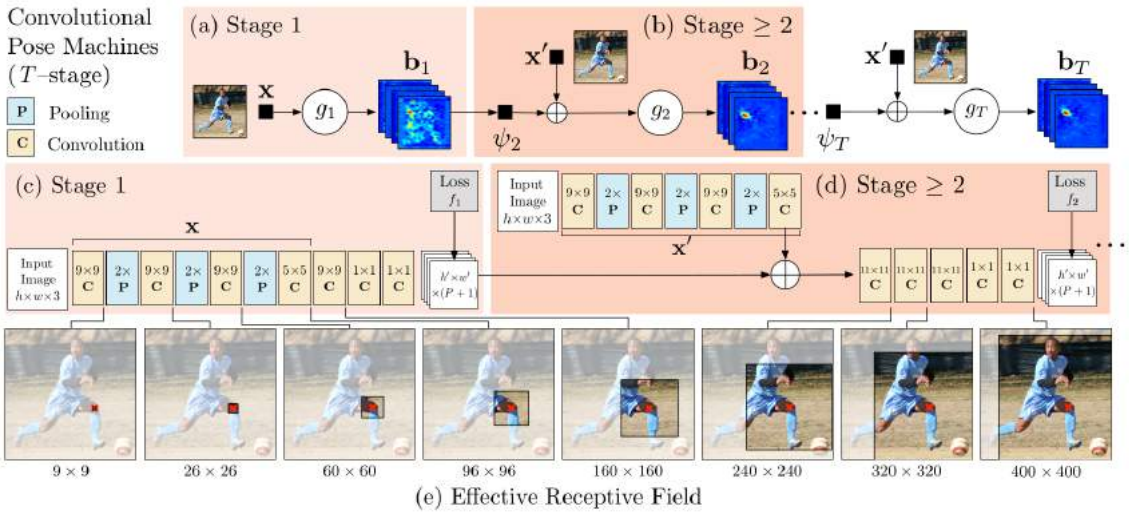


Figure 8: Overall architecture of the body part detection network. Image taken from [20]. (a), (b) sequential stages of the pose machines approach. (c) and (d) sequential architecture of the convolutional pose machine. Each stage produces belief maps for the position of body parts in the image taking into account belief maps of previous stages and image features. (e) Effective receptive field on the original image.

refine the belief maps. Figure 8(c) and (d) shows the integration of contextual information $\boldsymbol{\psi}_t(z, \mathbf{b}_{t-1})$ of belief maps from the previous stage and image features $\mathbf{x}$ as the input to stage t.

Real Time Multi-Person Detection. Currently we use realtime multi-person convolutional pose machine framework presented in [4]. In this approach, both the body parts and limbs detections are performed jointly in a two-branched-network. Following a similar approach of the single-person network, the network architecture shown in figure 9 adds two branches for each stage t. One branch produces belief maps of body parts and the second branch produces belief maps of limbs connecting pairs of body parts of different type, e.g. left shoulder and left elbow.

An example of the pipeline outputs is shown in figure 10. The figure 10(b) shows the belief maps of body parts meanwhile figure 10(c) shows the belief maps of the limbs that connect body parts of different types. It's worth to mention that the production of belief maps of body parts and limbs is performed jointly and not sequentially.

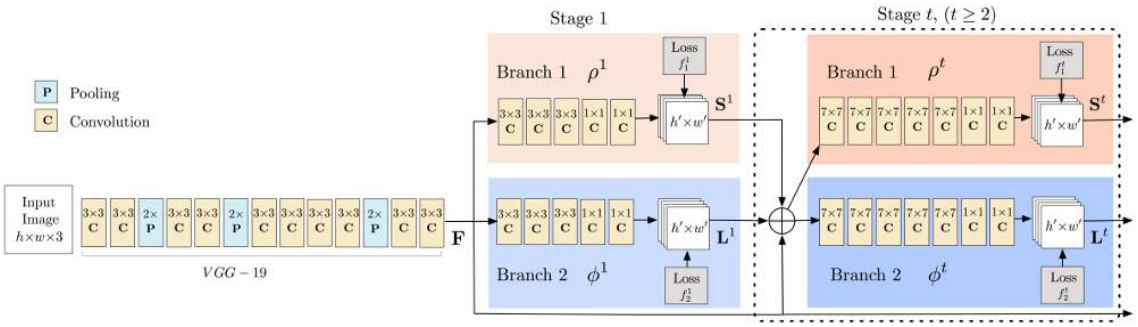


Figure 9: Network architecture of real time pose estimation approach. Image taken from [4]. The two-branched-network architecture inherits the sequential procedure for belief maps production and divides the sequence into two branches that produce belief maps for body parts and limbs.

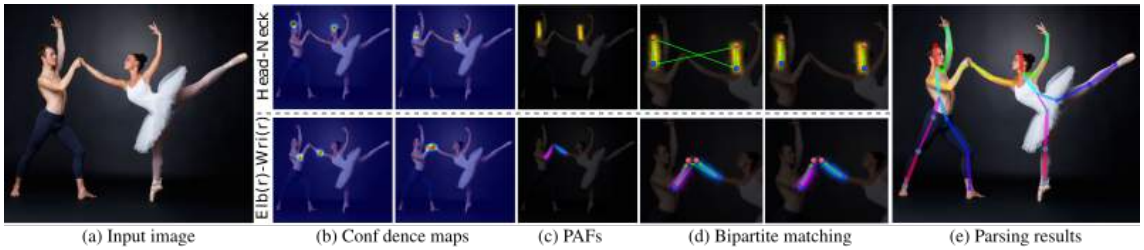


Figure 10: Overall pipeline for multi-person pose estimation. Image taken from [4]. (b) and (c) show belief maps of body parts and belief maps of limbs respectively. (d) and (e) show the keypoint association to form limbs and the final constructed full body poses.

5.2.3 Improvements and integration

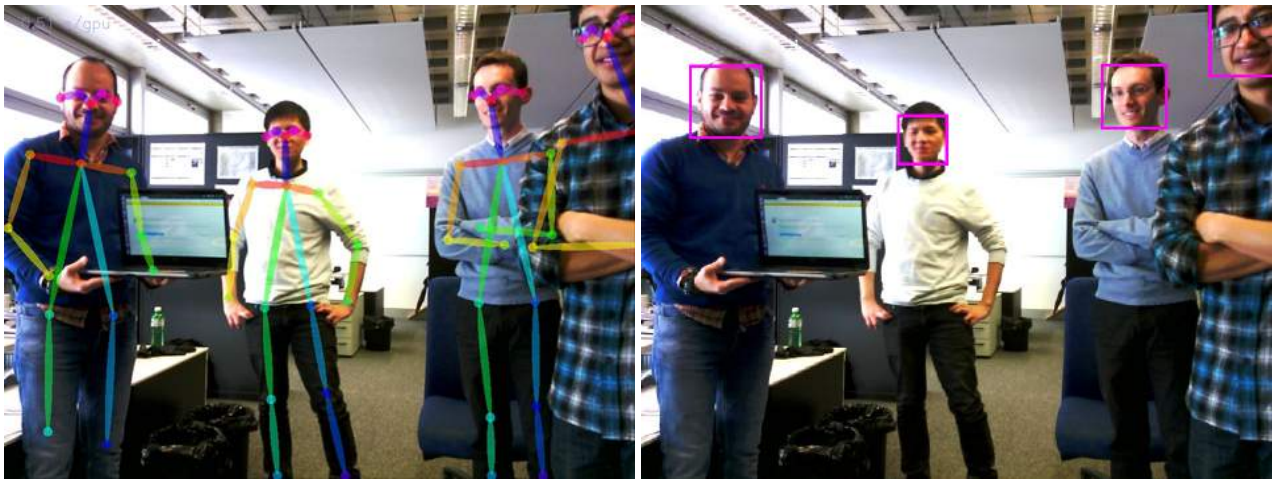
In this section we describe our work to evaluate CPM and combine it with our tracker. The main goal is to leverage all the nice properties of the head tracker described in section 5.2.1 (processing speed, NVB extraction, attention, nod, etc.), and to make it more robust by using the CPM estimator as initial face detector.

Setup and Experiments We use the implementation released by the authors found at [1]. The learnt model is built from training with the MSCOCO dataset [2] which contains over 1M keypoint annotations for 100K people in RGB images and is suitable for the training of deep convolutional neural nets. The realtime performance of the framework is subject to GPU availability. Currently, our setup consist of a laptop with NVIDIA Quadcore M2000M GPU card. Common outputs of the system, on video recorded from Pepper, are visualized on figure 11.

The network setup provided by the authors runs at 0.5 fps on our laptop. Nevertheless, reducing the network resolution improves the running time performance up to 7.7 fps at the compromise of losing few accuracy on the body detections.

Face Tracking and Convolutional Pose Machines The convolution pose machine processes the input stream frame by frame without the notion of tracking. As explained in section 5.2.3, it provides the position of the human

Figure 11: Output of the system of video recorded from Pepper. The images show the articulated poses overlayed on the image containing multiple people.



(a) Convolutional Pose Machines output (b) Inferred heads from CPM output

Figure 12: The position and the size of the heads (figure b) are inferred from the parts provided by the CPM detector (figure a) based on the position of the nose, ear, and eye parts.

parts (eyes, nose, elbow, etc.) of all the detected persons in the input frame (see figure 12a).

To leverage the accuracy of the CPM estimator as well as the efficiency of the tracker presented in section 5.2.1, we replaced the original OpenCV head detector by the face detected by the CPM detector. The head detection is estimated based on the joints of the eyes, the ears, and the nose (figure 12b). The tracker therefore initialises the tracks with the face detection of the CPM.

Figure 13 shows the output of the combination of the CPM and the tracker. We can see that all faces are detected (the 3 purple squares for 3 heads) but they are shifted from the current position of the head because of processing delay. In the background, we can see that the person is detected (small purple face) but that it is not tracked yet. This image illustrates two important points:

- The CPM detector is efficient in detecting humans (even small ones in the background); it also gives good results on the VGA RGB images taken from Pepper.
- The processing time is a bottleneck because we get the results of the detection a few frames later.

Replacing the NAOqi tracker. Our multi-person tracker with CPM replaces the “PeoplePerception” part of NAOqi, but we still need the output of the “FaceCharacteristics” from NAOqi. Hence a fusion between the two independent modules is required. The fusion is done according to figure 14. The results of the two independent modules (face tracks, and gender, smile, expression from NAOqi) are linked given the position of the detected head in the image. Since the CPM has better detection capacity, our tracker may follow more persons that NAOqi can detect. In this case, the published messages have empty fields.

Improving the processing speed. We are currently investigating several strategies to improve the speed of the CPM. One way presented earlier was to reduce the resolution of the network at the cost of lower localisation of the body parts. Another strategy, depicted on figure 15, consists in tiling several images into the same one, and to apply the CPM on this larger images. Since the CPM internally resizes the input image, this does not affect the processing speed. Regarding the accuracy of the CPM, the only drawbacks are:

- very small person are not detected. However, this is unimportant in our case, because in practice, this corresponds to people standing more than 15 meters away from the robot.
- localization accuracy degradation.
- a more important issue of this method is that, if frames are processed 4 by 4 (instead of 1 by 1), the algorithm has to wait for 4 (continuous) frames to be available before processing them, which cause the

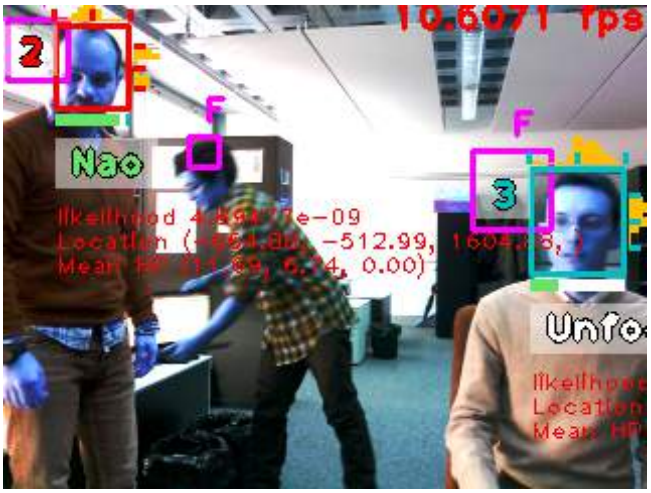


Figure 13: Output of the tracking algorithm on an image from Pepper. Non-purple squares are the tracked persons; the number are the track ID of the persons. The text below the tracks indicates whether the person is looking at the robot (“Nao”) or not (“unfocused”). Purple square are face detection which are not turned into tracked. On this image, there are shifted compared to the actual face because of the detection delay between the CPM and detector.

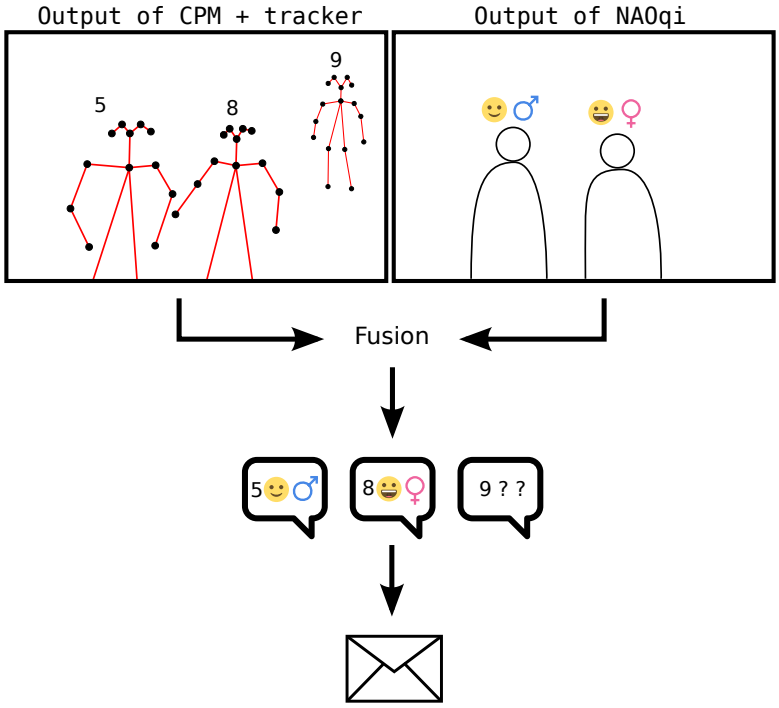
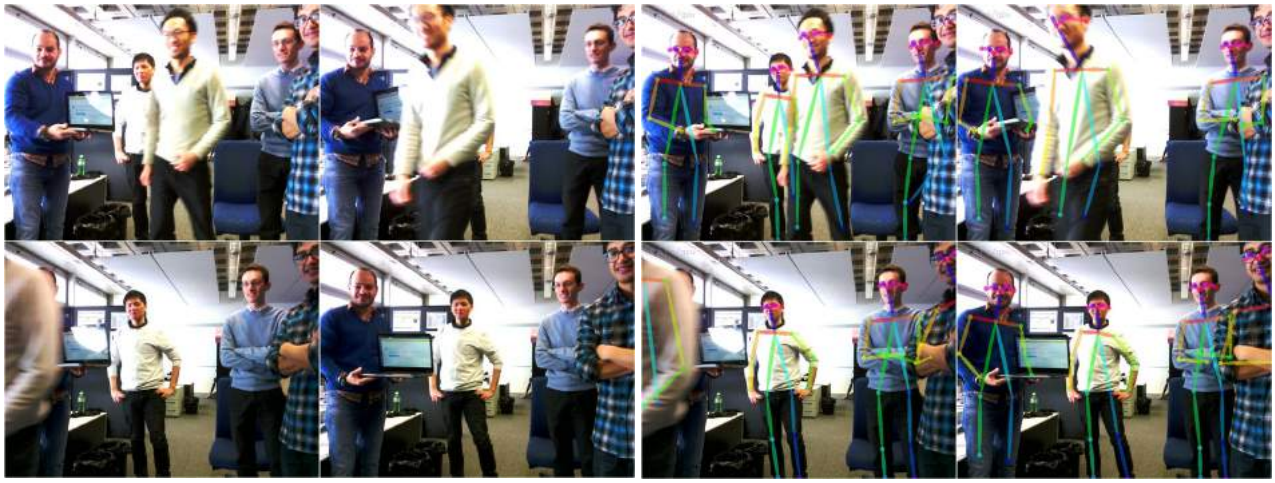


Figure 14: Fusion of the “FaceCharacteristics” module from NAOqi and of our head-pose tracker. The facial expression, the smile, the gender, and the age of a person are directly taken from the “People Perception” module from the NAOqi driver. while the face localizations come from our CPM-based tracker. The fusion consists in merging the two independent results into the same ROS messages, and assign the facial expression from NAOqi to the head locations of our tracker.

first one to be processed later than if it was 1 by 1. But this allows to have more processed frames for the tracker.

Future developments in designing and training our own architecture will address these issues.



(a) 4 concatenated input images into a bigger one (b) Output from CPM on the concatenated input

Figure 15: The CPM detector speed is independent of the size of the input image because the latter is resized to a default size. Concatenating several images into one enables to process multiple images at the same time without hurting the performance (only very small person would not be detected). However, it requires to wait for 4 images to be available instead of online processing.

5.2.4 Conclusion

The investigation lead on the tracking part show that the performances are robust, especially thanks to recent state-of-the-art neural-network-based methods for detection of body parts. The tracking aspect on top of the body part estimation will be ready soon, and available for partners to improve the perception part of their scenarios. An important point that will be discuss with partners for the integration, is the hardware (GPU), to be able to run the GPU-based perception module.

The preliminary experiments shows that it will be important to adapt the current model on the following points:

- The system needs to be fully real time and should avoid any latency. We will investigate how to retrain such models with a view to fully integrating it inside the perception module.
- A good re-identification module is off the essence for the scenarios to be fully operational. We can expect this problem to be handle by the good performance of the CPM.
- We can expect to get a real 3D pose (as opposed to a 2D estimation) as soon as the RGB and depth are correctly synchronised.

6 Perception: Close/medium range person sensing

For people within 0.5 to 1.3 meters and whom Pepper is interacting with, improving the accuracy of tracking and of non-verbal behavior sensing (gaze, head or body gesture, expression) is important. In particular, it can benefit from the availability of good depth data which is useful for tasks requiring 3D analysis, such as 3D gaze i.e. line-of-sight- estimation.

In this Section, we report the work done in Y1 along this line. First, we summarize in Section 6.1 the new methodology developed during Y1 for 3D head pose tracking using depth data, which provides much more robustness for tracking under adverse poses. Then, in Section 6.2 we report on the other progresses made this year beyond robust pose estimation, centered on code optimization, gaze and head gesture integration, and research on gaze improvements.

6.1 Head pose estimation

6.1.1 Motivation

To achieve accurate 3D head and pose tracking, methods relying on registering a 3D face mesh potentially learned online to depth observation sequences have achieved the best performance so far [15]. They often use 3D Morphable Models (3DMM) as mesh representation, since they provide a low dimensional representation of the shape allowing user-specific well constrained model adaptation.

However, a limitation of most 3DMM models is that they do not contain the top, side and back parts of the head because it is actually quite difficult to extract linear statistical basis from the variations of the hairs (and even the ears) in these parts. In many scenarios, however, being able to track the head even under more profile or extreme poses is required to avoid tracking interruptions or failures and manage such poses as encountered in natural setting like the ones planned in MuMMER (e.g. when giving directions to people, they may look away from Pepper).

To address this issue, we have investigate a method for robust and accurate head pose estimation. The method relies on two main steps performed jointly: the online reconstruction of a full 3D head model, which is based on a variant of KinectFusion. and online fitting of a 3D morphable model by selecting multi-view observation samples and rendering smoothed synthetic samples. By combining the strengths of these two models in a single representation, we show that both accurate and robust performance can be obtained even under very extreme head poses. Our method achieves state-of-the-art results in the benchmark BIWI dataset [7].

The work as been published in the Face and gesture conference 2017 [22], and details can be found there.

6.1.2 Approach overview

The proposed framework is illustrated in Fig. 16. It consists of three main modules: head pose estimation, 3DMM fitting and head reconstruction. The pose estimation module aligns at every time step i the current head model $\mathbf{h}^i$ with the observed depth map data $\mathbf{o}^i$ using a variant of the Iterated Closest Point (ICP) algorithm. The aim of the two other modules is to learn and update the head model $\mathbf{h}^i$ of the given person using the sequence of observations. This is achieved using two main representations: the first one, $\mathbf{r}^i$, is a 3D reconstruction of the head obtained through the temporal registration and integration over time of the incoming depth frames. Its main advantage is that it can represent the full head without any prior knowledge. The second one is a 3DMM face representation, $\mathbf{m}^i$, built and adapted online using a 3DMM fitting algorithm relying on automatically selected depth frames, complemented by synthesis data from the reconstruction model, whenever it becomes available. The resulting head model used for pose estimation if thus given by a set of vertices coming from the two representations, $\mathbf{h}^i = \{\mathbf{m}^i, \mathbf{r}^i\}$.

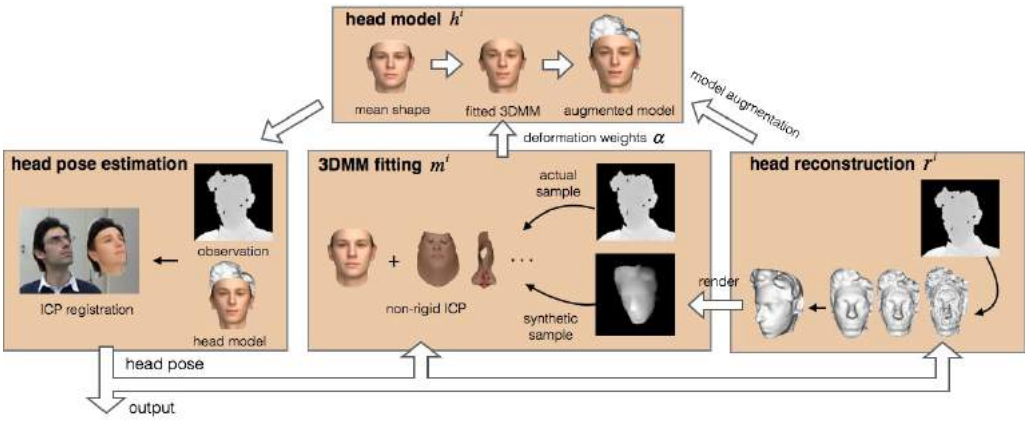


Figure 16: Proposed framework. At time i, the head pose estimation module registers the current head model $\mathbf{h}^i$ to the observations. The 3DMM fitting module personalizes a 3DMM face model $\mathbf{m}$ to sample frames and their synthetic version generated from the head reconstruction. The reconstruction module aggregates pose rectified depth images into a full head representation $\mathbf{r}^i$. Vertex samples from the 3DMM face model $\mathbf{m}^i$ and from $\mathbf{r}^i$ are used to define the head model $\mathbf{h}^{i+1}$.

Figure 17: Results of head pose estimation using only the 3DMM (top) and augmented model (bottom).

Note that both face 3DMM and head reconstructions are built online without any manual intervention.

6.1.3 Results and conclusion

Results. Our method achieved the best results on the BIWI public benchmark dataset comprising 24 people sequences. In addition, it provided very robust method on natural interaction data involving natural behaviors of humans. Sample results are shown in Fig. 17.

In addition, results showed that the use of both models was useful for accuracy (provided by the 3DMM) and robustness (full head). In addition, keeping the 3DMM allows to have semantic information (e.g. about eye localization), which is essential for further face analysis modules like gaze (see the next section).

Computational cost. While we are now able to process the 3DMM tracking in real-time (see progresses this year, Section 6.2), the current reconstruction approach can run at around 5 fps on CPU, but could benefit form GPU acceleration and further optimization.

6.2 Tasks progress

Different works have been done towards (i) making the medium perception modules within the project framework running in real-time, and (ii) improving their performance. We report on them below.

6.2.1 Code optimization

Our head pose and eye gaze tracking methods build upon the RGB-D and HG3D frameworks[7]. This framework is able to run at 8 frames/second by default on a standard computer when it comes to the head pose tracking alone. This is not sufficient for the following reasons: i)if the change on the head's pose is too large between two successive frames, the algorithm converge to a local minima far from the true head pose, as it is common under low frame rate conditions; ii) the resulting sampling rate for the head pose parameters may be insufficient for head gestures detection algorithms to detect subtle head movement patterns, as it is the case for head nods.

Therefore, to improve both elements, we modified the software in order to obtain 30fps for the head pose tracking algorithm. This has been validated through connected devices, such as the Asus Xtion Pro Live, Primesense Carmine 1.09 and the Kinects 1 and 2. QVGA depth resolution is preferred as it offers a good trade-off between computational costs and head pose accuracy.

6.2.2 Integration

We combined the modules of head pose tracking, 3D gaze estimation, and head nods detection into a single system running real-time, facilitated through the optimisations mentioned above. The head pose tracking approach is the same method described in [10] extended to capture a person specific 3D face model on-the-fly. The gaze tracking algorithm consists of a support vector regression method trained from multi-scaled histograms of oriented gradients features employing an eye appearance rectification approach for head pose invariance, as described in [10]. The head nod detection algorithm consisted of using pose invariant angular features over a time-window to train a support vector classifier, described in [5].

At this point, the system runs with an RGB-D consumer device connected to a laptop. We have also integrated NAOqi to use the data captured directly from Pepper's sensors. However, the algorithms results are not satisfactory due to the poor depth data quality encountered so far, as described in Section 4.1. It is yet pending to do further tests using the depth data coming from recent improvements but not available yet at Idiap (c.f. Section 4.1) as well as to package this software as a ROS node.

6.2.3 Deep gaze estimation

Within the methodology of appearance based methods, which we explored in [10], we have investigated the used of deep convolutional neural networks (CNN) for the prediction of the eye gaze direction from eye image data.

We have implemented the approach described in [23], consisting of a sequence of 2 convolutional and max-pooling layers, together with a fully connected layer which combines both the extracted eye features with the head pose parameters. Instead of including the pose parameters into the features set, we are using the eye appearance rectification method described in [10], which removes appearance variations due to head pose. The network is then trained using the EYEDIAP database [8].

Results can be seen in Tables 3 and 4. For a more detailed description of the diverse protocol for training and evaluation, and different methods, please refer to [10]. Consistently, the results using this deep CNN approach are lower than with previous methods. However, these are early evaluations and we expect improvements as we tune the training strategy, considering elements such as learning rate, and training data size. As well, all results consist of training a network from scratch with parameters initialised randomly. Other methodologies could employ the pre-training of the network with larger training sets, and then training the last lawyers with the specific task. This remains pending for future work.

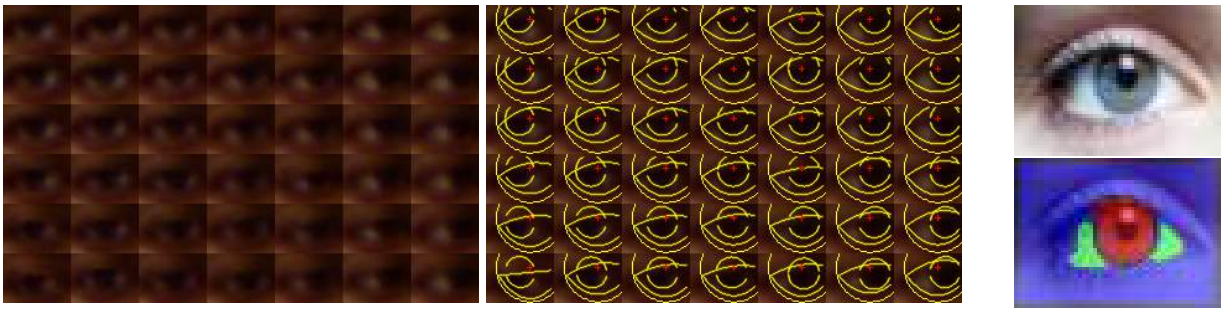
[7] Early versions of this software are available at `https://www.idiap.ch/project/g3e/data-and-softwares`

Table 3: Summary of results on mean angular gaze error (°) for the floating target conditions (*FT*) using the EYEDIAP database as an extension of the table presented in [10]. For a given experimental protocol we report, per evaluated method, the mean (top) and standard deviation (bottom) computed over all relevant sessions for the given conditions: i) *SP-PS*: static pose with person-specific gaze models ii) *MP-PS*: mobile pose with person-specific gaze models iii) *SP-PI*: static pose with person invariant model iv) *MP-PI*: mobile pose with person invariant model. Acronyms: *SP* (static head pose) - *MP* (mobile head pose). *PS* (person specific trained model) - *PI* (person invariant model). D (*DDM* data-driven rectification) - T (*TDM* template-driven rectification). *NA* (no eye image alignment) - *FL* (automatic eye corners detection based eye image alignment) - *EC* (manual eye corners annotation based alignment) - *A* (SDIPA-based supervised alignment) - *A5* (SDIPA-based supervised alignment using only 5 samples for the test subjects).

	SP-PS	*MP-PS* pose invariance		*SP-PI* person invariance					*MP-PI* pose and person invariance				
Method	-	D	T	*NA*	*FL*	*EC*	*A*	*A5*	*NA*	*FL*	*EC*	*A*	*A5*
HP	28.6	23.0	23.0	28.0	28.0	28.0	28.0	28.0	23.6	23.6	23.6	23.6	23.6
	3.0	4.0	4.0	2.7	2.7	2.7	2.7	2.7	3.9	3.9	3.9	3.9	3.9
kNN	6.4	9.8	9.8	12.2	11.9	10.9	10.0	**10.0**	13.7	13.4	13.3	12.2	12.4
	1.5	2.6	2.7	3.2	3.5	2.7	2.1	1.8	3.2	4.0	3.5	2.7	2.5
ALR	6.2	11.5	10.3	13.7	-	-	-	-	-	-	-	-	-
	1.9	2.2	2.0	4.6	-	-	-	-	-	-	-	-	-
H-SVR	6.0	9.3	9.0	11.8	**11.3**	10.7	**9.8**	10.4	13.0	12.6	12.0	11.6	11.8
	1.9	2.2	2.1	3.8	3.2	2.9	3.1	3.5	2.5	3.1	2.3	2.2	2.5
R-SVR	**5.6**	**8.5**	**9.0**	**11.6**	11.6	**10.6**	10.5	11.0	**11.7**	**12.4**	**12.0**	**11.4**	**11.6**
	1.7	1.8	2.7	5.1	3.5	3.4	3.6	3.7	2.7	3.2	2.6	2.4	2.4
CNN	7.91	10.54	-	13.64	-	-	-	-	13.88	-	-	-	-
	3.18	4.30	-	4.89	-	-	-	-	3.98	-	-	-	-

Table 4: Summary of results on mean angular gaze error (°) for the screen target conditions (*CS*). We report the mean (top) and standard deviations (bottom) computed over all sessions relevant for a given condition. For acronyms, see Table 3.

	SP-PS	*MP-PS* pose invariance	*SP-PI* person invariance					*MP-PI* pose and person invariance				
Method	-	T	*NA*	*FL*	*EC*	*A*	*A5*	*NA*	*FL*	*EC*	*A*	*A5*
HP	13.5	15.7	13.0	13.0	13.0	13.0	13.0	15.7	15.7	15.7	15.7	15.7
	2.9	4.2	2.5	2.5	2.5	2.5	2.5	4.2	4.2	4.2	4.2	4.2
kNN	2.9	4.2	8.7	7.8	7.6	6.6	6.6	9.8	9.0	8.6	7.6	7.8
	1.2	1.3	3.5	2.2	2.8	2.9	3.2	2.6	2.0	2.5	2.3	2.7
ALR	2.4	4.8	9.2	-	-	-	-	-	-	-	-	-
	0.9	1.7	3.9	-	-	-	-	-	-	-	-	-
H-SVR	1.9	**3.5**	**5.8**	**6.2**	**5.7**	**4.9**	**5.1**	**6.8**	**6.8**	**7.0**	**5.7**	**6.0**
	0.8	1.3	3.0	2.5	2.7	2.2	2.1	2.6	1.9	2.2	1.9	2.1
R-SVR	**1.7**	3.6	6.6	6.4	6.6	6.0	6.6	7.6	7.1	7.3	6.4	6.9
	0.8	1.4	3.1	2.7	3.6	2.6	3.5	3.3	3.0	3.2	2.4	3.3
CNN	3.85	5.98	7.98	-	-	-	-	9.25	-	-	-	-
	1.9	2.65	2.7	-	-	-	-	2.57	-	-	-	-



(a) (b) (c)

Figure 18: Examples of data and results for the G^{3}E methodology. Figure (a) shows a training set of eye images obtained from the EYEDIAP database. Note the poor illumination and contrast. Figure (b) shows the resulting geometric model and configuration which explains best the data according to the G^{3}E approach. The color distributions are known a priori for this case. Figure (c) Example of automated eye segmentation for the purpose of color distribution learning, as a first step for the G^{3}E methodology.

6.2.4 Geometric generative gaze estimation (G^{3}E) improvements

As described previously, the G^{3}E approach combines the advantage of geometric based methods with appearance based methods through a generative probabilistic model. Two main challenges remain, which we have addressed as follows.

Model inference. Our proof-of-concept paper [9] relied on variational Bayes combined with numeric approximations for the objective functions' Jacobian through sampling techniques. We found this approach nevertheless limiting, as the numeric approximations' render the problem as a stochastic optimization, making it difficult to employ standard gradient-based optimization methods. As a result, it becomes difficult to reach the global minima, leading to gaze estimation errors.

Recently, we are employing MCMC inference methods, where we approximate the sampling of the posterior distribution through a moving kernel sampling distribution (which is part of the MCMC methodology). Note that the parameters space correspond to the parameters of the eyeball geometry. We found this approach to be more reliable in terms of the final estimate, although it is computationally costly, as in a training phase we are employing samples in the order of 5000 to 10000 samples, each of which require an image likelihood evaluations on the training set. We are working towards finding faster inference strategies. An example of training set and result can be seen in Figures 18a and 18b.

Color distributions learning. The G^{3}E approach, as described, relies on the segmentation of the eye image into three semantic regions: the skin, sclera and the cornea (iris+pupil region). Our initial work relied on the manual annotation of pixels from each of these regions in order to fit color distributions representing the appearance of each region. We have therefore proposed strategies to learn these color distributions automatically.

The approach consists of a series of heuristics leveraging the expected position and orientation of the eyeball (given that the eye direction is known at training time). In short, the algorithm proposes a series of candidate positions for the cornea region, which are expanded through the watershed algorithm. From this plausible location of the cornea, side pixels are obtained from the sclera and the region is also expanded using the watershed algorithm. The skin (eyelids and surroundings) region is obtained from picking color pixels from the surroundings of the eye region, which can be obtained reliably thanks to the person-specific face model fitted through the head pose tracking method. Finally, a segmentation is chosen from the resulting segmentation of each candidate through another set of heuristics, i.e., the sclera region is always "more white" in average than the cornea region. An example can be seen in Figure 18c. This approach is yet to be evaluated extensively on the EYEDIAP dataset.

7 Perception: Human Activity recognition

For a robot working in a human environment, it is important to be able to detect people. Once the robot starts interacting with a particular person, it would be also advantageous to analyse the person's state such as mood, emotions, and also activities. Thus, SBRE is working on visual perception of human activities. The objective is to recognize human activities based on data acquired from embedded sensors on Pepper.

7.1 Methodology

For the moment, our approach for semi-supervised activity recognition is based on RGB images and the system is trained on a dataset of labelled RGB images such as Action40 [21] describing people performing different actions such as drinking, cooking, calling, etc. The trained system allows to recognize human activities that present in the dataset, and during recognition, we compute the saliency of the recognized class and localize it in the image space. Our approach is based on [24] adapted to recognize and localize human activities based on a single frame.

7.2 Results

We are currently evaluating our approach on public databases, such as Action40 (see Fig. 19), and also on our own image dataset recorded from sensors embedded on Pepper (see Fig. 20b).

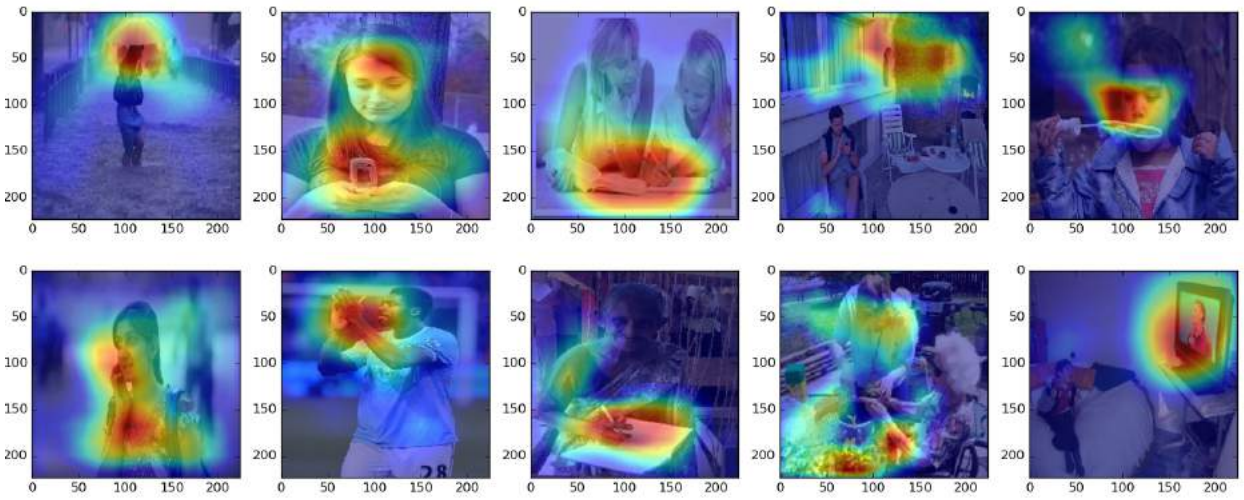


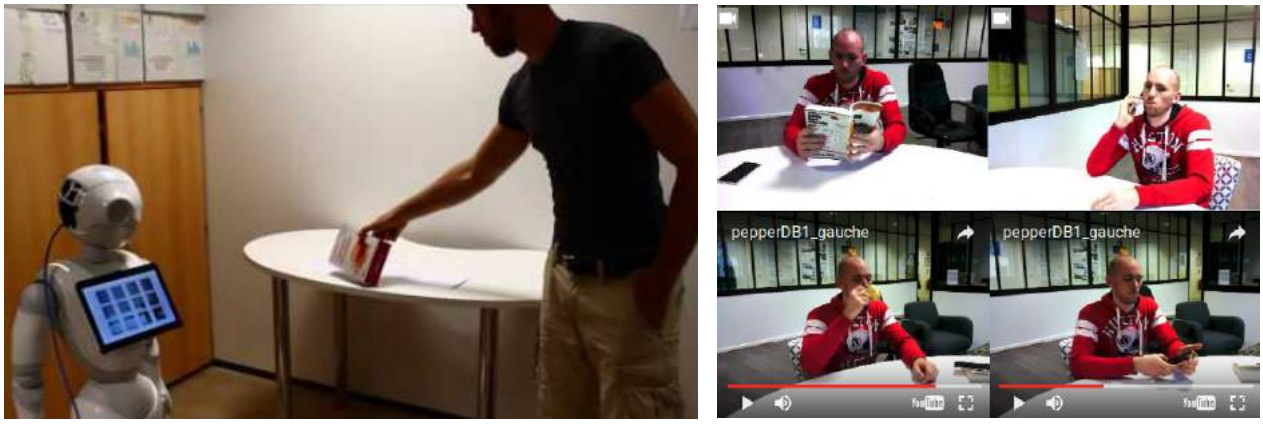
Figure 19: Example of recognition and localization of human activities on the publicly available dataset Action40; the color scale represents the probability of localizing a certain class at each position on the image (both positive and false-positive examples of recognition are shown)

7.3 Conclusions

We have already done initial integration of our approach for human activity recognition based on RGB images on a robot (see Fig. 20) where the visual data are acquired from an RGB camera embedded on Pepper and the data processing is performed on a remote GPU. Any hardware configuration described in Section 3 (see Fig. 2) with either remote or embedded GPU would be suitable for our system.

7.4 Outputs and Future Directions

The detailed results on evaluation will be provided in the next reporting period. We also consider to extend our approach to recognize human activities based on a sequence of RGB frames acquired over time, and fusing with 3D data and audio data.



(a) The setup with a human showing an object to Pepper; the results of a class recognition and localization are shown on the Pepper's tablet (12 most probable classes are shown)

(b) The database of human activities (here: reading, calling by phone, drinking, reading messages) that was recorded on Pepper and used for evaluation

Figure 20: Integration on Pepper and examples of images from the database recorded on Pepper and used for evaluation in addition to publicly available databases

8 Perception: 3D skeleton tracking

SBRE has been working not only on detecting people and their faces but also on detecting his/her poses, gestures, and actions. This will be useful to achieve a better understanding of a person interacting with the robot. Thus, SBRE is working on skeleton detector allowing to detect the position of a head, shoulders, elbows, and hands of a human in front of the robot. To perform this detection we are relying on depth data from the embedded depth sensor on Pepper.

8.1 Methodology

The solution proposed is inspired by the SkelTrack software which has been release on an open source licence. The general scheme is as follow:

- The cloud of points returned by the camera is down-sampled, and a graph is created.
- The graph is explored so as to find three extremities: the head, and the hands
- Based on these extremities the position of the shoulder and elbows are computed.

The general scheme of the original solution has been kept but the code rewritten so that it can run on SBRE's robots. Modifications to the original scheme have been made to:

- improve the computation of the extremities (head and hands) as well as the deduction of the position of the articulation (shoulders and elbows).
- avoid the return of abnormal position of the joints.
- allow the return of joints when hands are outside of the field of view of the robot.
- allow the return of joints when only one part of the skeleton has been detected.

Based on all these modifications the project has been renamed SkeletonDetector.

8.2 Results

SkeletonDetector algorithm takes as input the depth camera of Pepper and is able to run at 3 fps. SkeletonDetector is efficient as long as part of the top-body is visible as illustrated in Figure 21.

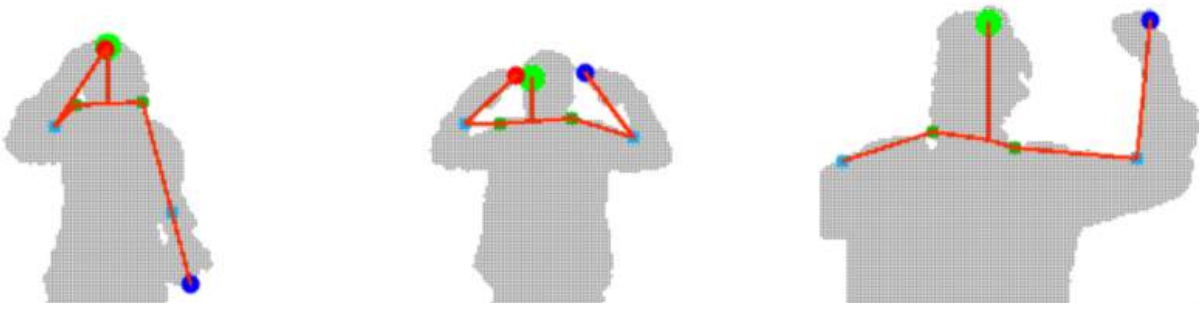
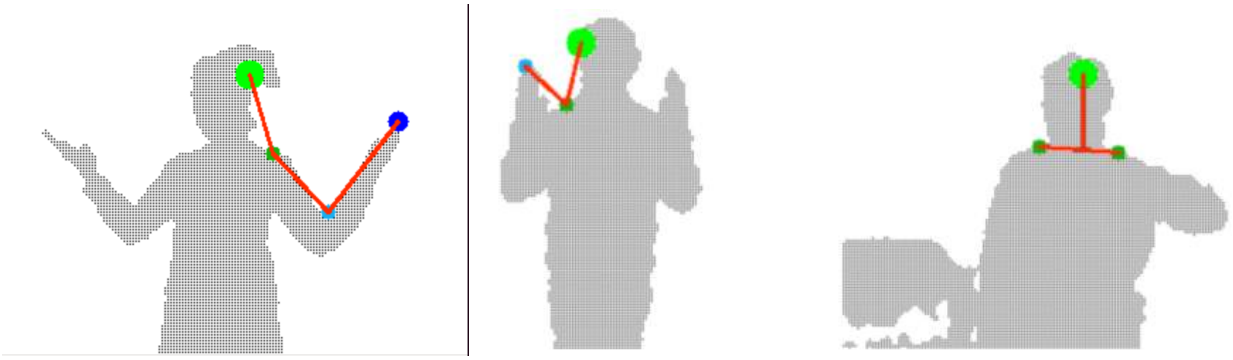


Figure 21: Examples of working cases for SkeletonDetector.

However, when the body is not facing the camera (see Figure 22a) or when part of the body is not visible (see Figure 22b), the performances of SkeletonDetector are strongly reduced.



(a) Body not facing the camera

(b) A part of the body is invisible

Figure 22: Example with bad orientation of the body.

8.3 Conclusions

The current version of SkeletonDetector is able to provide basic information on the position of users head, shoulders, elbows, and hands. These informations are estimated to be sufficient and reliable enough to be use in scenarios where a user point in a direction.

The source code has been released as open-source and SBRE is welcoming participations.

8.4 Outputs and Future Directions

The project has been developed and evaluated on Pepper as it is (without any modification of the depth sensor and its driver, and with the current version of lenses in front of the sensor), thus it faces limitations due to sensor data quality, Pepper computational power, strategy used to detect the user's position. We are planning to continue to work on the general strategy used so as to improve it. This may require a change of paradigm.

9 Perception: Audio sensing

Sensing whether someone speaks, and whom, is important for HRI systems. We have started investigated this topic with deep neural networks (DNN), since there are several interesting arguments in favor of using them.

As this is an ongoing work, we only summarize below some main elements. More will be done during Year 2 on that topic.

9.1 Introduction

Audio sensing is of great importance for human robot interaction. To achieve natural interaction it is crucial for the robot system to understand who is talking and what is said. These tasks can be very challenging, especially when the interaction involves multiple persons in an unconstrained environment, like a shopping mall. In this project, we try to tackle the audio sensing issues for practical HRI.

As the first step, we plan to develop robust sound source localization (SSL) algorithms targeted for Pepper to localize speech-type audio source. Most previous SSL methods (for example GCC-PHAT, SRP-PHAT, etc.) require strong assumptions about the environment, such as that sound travels directly from source to microphones, there are little or no reverberation, and not too much ambient noise, etc. However, these assumptions hardly hold in reality.

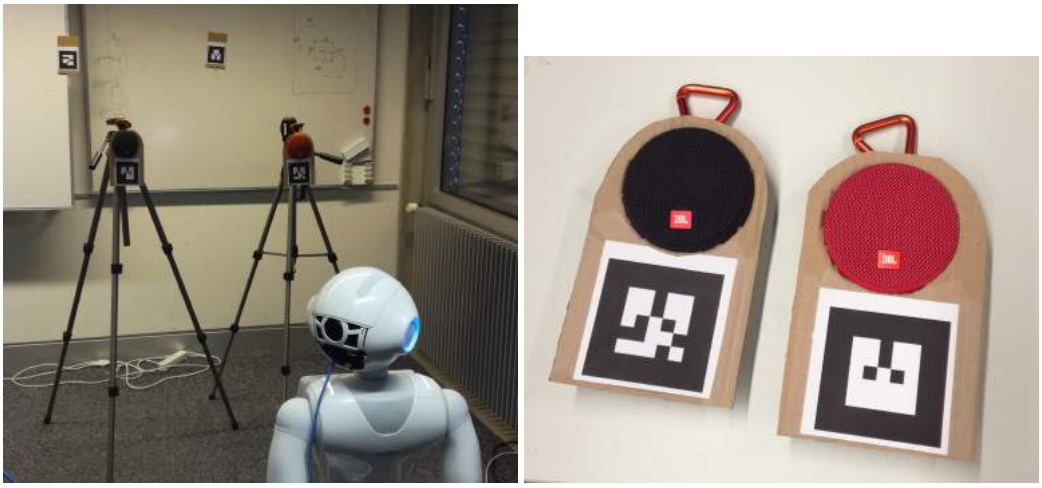
To tackle these problems, we will investigate learning methods which require minimal assumptions but may require a relatively large amount of trainign data. The learning methods should learn to directly associate input signals with sound source directions in various environments. With DNN, it should be possible, amongst other, to automatically learn representations that are robust to changes in environments.

9.2 Data Collection

Training of the learning methods requires sufficient amount of data. Unfortunately, there does not exist any HRI dataset available at the moment. In particular, there exist no dataset recorded with Pepper. Therefore, we have worked on preparing a framework allowing to do the automatic collection of a large amount of data by ourselves, so that we can develop novel SSL methods and do experiments.

We have developed a system to semi-autonomously record artificial data. We use loudspeakers to play clean speech segments from natural interactions and record from Pepper (see Figure 23a). The ground truth of sound source location is automatically acquired by detecting markers attached to speakers (see Figure 23b). The robot can move its head and torso so that the recording can cover a large range of azimuth and elevation source directions. Data will be recorded in different rooms, so the dataset will cover different acoustic environments. In addition, background noises will be recorded as well and used to simulate different ambient noises.

The recording system has been finalized, and the first data collection phases are ongoing.



(a) Recording environment. (b) Loudspeakers with markers.

Figure 23: SSL data collection.

10 Conclusion and future plans

The work performed within work package 2 during the first year has yielded many results that will be useful in the rest of the project. An important result of the reporting period are the WP2 output **API specification** and its implementation, which have been tested and used by other modules and partners, and will simplify further upgrades during the whole project. Other results involves the ongoing integration of multi-person tracking framework delivering further non-verbal (NVB) cues, like attention and head gestures, and which relies on a robust head and body detectors (see figure 13) based on a recently published state-of-the-art body part estimator (see figure 12a); a robust 3D head pose tracker based on RGB-D data for which a simplified implementation runs in real-time; and other elements about human activity analysis and gaze estimation. Besides the overall WP2 objectives, Idiap and SBRE have closely interacted (tests, recordings, discussions) to improve the quality of the depth sensor, which lead to a decent quality of the depth image so far.

Altogether, WP2 is on good track to reach the MS5 milestones "Software components ready for integration on robot, for month 24" which will be useful for performing more thorough user studies. To achieve this goal, the following lines of research are ongoing or planned:

- Multi-person tracker: we plan to investigate more thoroughly the human body part localisation estimator (section 5.2.3), possibly improve it and speeding it by taking as input the depth image in addition to the RGB image, or to detect object of interest in addition to persons. In parallel, a main focus for year 2 will be on working on the person re-identification part, to much improve the capacity at keeping people IDs throughout the interactions.
- Human activity recognition: we consider to extend our approch to activity recognition based on fusion of visual and audio data and also taking into account visual data over time rather than one-shot recognition.
- Non-verbal behaviour: we aim to increase the robustness and accuracy of the gaze estimation with improvements on the G^3E methodology as well as through adaptation mechanisms from context (scene geometry, conversation flow, etc.). This also depends on proper RGB and depth synchronization which may require hardware improvements.
- Audio: sound localisation is going an important topic to precisely identify which person is speaking to the robot. The deep net approach sketched in previous Section will be implemented in the coming months, and should lead to appropriate results.
- Benchmarking: an important part will be to assess the improvements made during the project by careful and systematic benchmarking of baseline methods and improved methods. This will include the design of recording sessions in context, annotations, and evaluation of the current system and of the all the following implementations.
- Research: as the first year was more dedicated to have a functional baseline for all the partners, the following years will be dedicated to investigating new methods and improving state-of-the-art methods, which will be published in journals and conferences.

A API: ROS messages definition

The current section gives more detail on the ROS messages which are sent by the perception module described in section 3. The action modules (dialogue, motion, interaction) subscribe to one or several topics containing the output of the perception module in order to make decision on which person to interact with, and what to say.

The following message definition may evolve in time as we improve the tracker or depending on other partner requests.

```
uint32 person_id

uint32   track_id
duration track_age

# A person may be given several IDs as new tracks are created. When we
# realise that 2 IDs are actually the same person, we store the other
# IDs in alternate_ids. The rule to set the person_id given the
# several IDs is
#
#   person_id = min alternate_ids
uint32[] alternate_ids

float64  head_distance
bool     is_in_fov
bool     is_occluded
bool     is_face_detected
bool     is_head_detected

geometry_msgs/Pose2D body_pose_gp

geometry_msgs/Pose head_pose
```

Listing 1: TrackedPerson message

```
std_msgs/Header header

wp2_msgs/TrackedPerson[] data
```

Listing 2: TrackedPersonArray message

```
uint32  person_id

# When a person reappears, the track ID is different than the previous
# track ID.
uint32 track_id

# Read from PeoplePerception/Person/<ID>/AgeProperties
int32   age
float32 age_confidence

# Read from PeoplePerception/Person/<ID>/GenderProperties
int32   gender
float32 gender_confidence

# Read from PeoplePerception/Person/<ID>/SmileProperties
float32 smile_degree
float32 smile_degree_confidence

wp2_msgs/ExpressionProperties expression_properties
```

Listing 3: FaceInfo message

```
std_msgs/Header header

wp2_msgs/FaceInfo[] data
```

Listing 4: FaceInfoArray message

```
# Face attributes read from
#
#   PeoplePerception/Person/<ID>/ExpressionProperties
#
# in ALFaceCharacteristics

float32 neutral
float32 happy
float32 surprised
float32 angry
float32 sad
```

Listing 5: ExpressionProperties message

```
uint32 person_id

# When a person reappears, the track ID is different than the previous
# track ID.
uint32 track_id

uint32             head_gaze_available
geometry_msgs/Pose head_gaze

uint32             eye_gaze_available
geometry_msgs/Pose eye_gaze

# PeoplePerception/Person/<ID>/IsLookingAtRobot
float64 probability_looking_at_robot

float64 probability_looking_at_screen

# The (known) targets of interest a person can be looking at
wp2_msgs/IdWithProbability[] attentions
```

Listing 6: GazeInfo message

```
std_msgs/Header header

wp2_msgs/GazeInfo[] data
```

Listing 7: GazeInfoArray message

```
# Potential targets includes:
#  - persons (track_id or person_id)
#  - objects
#  - predefined directions (GAZE_LEFT, \etc)

uint32  target_id
float64 probability_looking_at_target
```

Listing 8: IdWithProbability message

```
geometry_msgs/Quaternion direction
```

```
float64 level
```

Listing 9: SoundSource message

```
std_msgs/Header header

wp2_msgs/SoundSource[] data
```

Listing 10: SoundSourceArray message

```
uint32  person_id
uint32  va_id
bool    is_speaking
float32 is_speaking_confidence
bool    is_last
time    turn_duration
time    last_turn
```

Listing 11: VoiceActivity message

```
std_msgs/Header header

wp2_msgs/VoiceActivity[] data
```

Listing 12: VoiceActivityArray message

References

[1] https://github.com/ZheC/Realtime_Multi-Person_Pose_Estimation.

[2] Common objects in context. http://mscoco.org.

[3] Sileye O Ba and Jean Marc Odobez. A Rao-Blackwellized Mixed State Particle Filter for Head Pose Tracking. In *ACM-ICMI Workshop on Multi-modal Multi-party Meeting Processing (MMMP), Trento Italy*, pages 9–16, 2005.

[4] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1611.08050*, 2016.

[5] Yiqiang Chen, Yu Yu, and Jean-Marc Odobez. Head nod detection from a full 3d model. In *Proceedings of the ICCV 2015*, pages 528–536, December 2015.

[6] S Duffner and J.-M. Odobez. A Track Creation and Deletion Framework for Long-Term Online Multi-Face Tracking. *IEEE Transaction on Image Processing*, 2013.

[7] G. Fanelli, J. Gall, and L. Van Gool. Real time head pose estimation with random regression forests. In *CVPR*, pages 617–624, June 2011.

[8] Kenneth Alberto Funes Mora, Florent Monay, and Jean-Marc Odobez. Eyediap: A database for the development and evaluation of gaze estimation algorithms from rgb and rgb-d cameras. In *Proceedings of the ACM Symposium on Eye Tracking Research and Applications*. ACM, March 2014.

[9] Kenneth Alberto Funes Mora and Jean-Marc Odobez. Geometric generative gaze estimation (g3e) for remote rgb-d cameras. In *IEEE Computer Vision and Pattern Recognition Conference*, pages 1773–1780. IEEE, June 2014.

[10] Kenneth Alberto Funes Mora and Jean-Marc Odobez. Gaze estimation in the 3d space using rgb-d sensors. towards head-pose and user invariance. *International Journal of Computer Vision*, 118(2):194–216, June 2016. First online: 13 November 2015.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[12] Vasil Khalidov and Jean-Marc Odobez. Real-time multiple head tracking using texture and colour cues. Idiap-RR Idiap-RR-02-2017, Idiap, 2 2017.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[14] Shihong Lao and Masato Kawade. Vision-based face understanding technologies and their applications. In *Advances in Biometric Person Authentication*, pages 339–348. Springer, 2004.

[15] Gregory P. Meyer, Shalini Gupta, Iuri Frosio, Dikpal Reddy, and Jan Kautz. Robust model-based 3d head pose estimation. In *ICCV*, December 2015.

[16] J.-M. Odobez, D Gatica-Perez, and S Ba. Embedding Motion in Model-Based Stochastic Tracking. *IEEE Trans. on Image Processing*, 15(11):3515–3531, 2006.

[17] E. Ricci and J.M Odobez. Learning large margin likelihood for realtime head pose tracking. In *IEEE International Conference on Image Processing*, november 2009.

[18] C Scheffler and J.-M. Odobez. Joint Adaptive Colour Modelling and Skin, Hair and Clothing Segmentation Using Coherent Probabilistic Index Maps. In *British Machine Vision Conference (BMVC)*, 2011.

[19] Ramakrishna Varun, Munoz Daniel, Hebert Martial, Bagnell Andrew J., and Sheikh Yaser. Pose machines: Articulated pose estimation via inference machines. In *ECCV*, 2014.

[20] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016.

[21] Bangpeng Yao, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas Guibas, and Li Fei-Fei. Human action recognition by learning bases of action attributes and parts. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1331–1338. IEEE, 2011.

[22] Yu Yu, Kenneth Alberto Funes Mora, and Jean-Marc Odobez. Robust and accurate 3d head pose estimation through 3dmm and online head model reconstruction. In *Proceedings of the 12th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2017)*, 2017.

[23] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. Appearance-based gaze estimation in the wild. In *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4511–4520, June 2015.

[24] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.