CSSR for

Culturally Sensitive Social Robotics for Africa

D3.1 System Architecture

Due date: **31/01/2024**
Submission Date: **24/01/2024**
Revision Date: **09/02/2025**

Start date of project: **01/07/2023** Duration: **36 months**

Lead organisation for this deliverable: **Carnegie Mellon University Africa**

Responsible Person: **D. Vernon** Revision: **2.1**

Project funded by the African Engineering and Technology Network (Afretec) Inclusive Digital Transformation Research Grant Programme		
Dissemination Level		
PU	Public	**PU**
PP	Restricted to other programme participants (including Afretec Administration)	
RE	Restricted to a group specified by the consortium (including Afretec Administration)	
CO	Confidential, only for members of the consortium (including Afretec Administration)	

Executive Summary

This deliverable represents the outcome of Task 3.1. It specifies the CSSR4Africa system architecture in detail, identifying the component subsystems, the modules (i.e., ROS nodes) comprising each subsystem, and the information exchanged between subsystems and modules. This includes a specification of the data that are input to each module, the data that are output from each module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed: through ROS topics, services, or actions, or through input and configuration files.

Based on the ROS nodes identified in Deliverables D2.2 and D2.3, Deliverable D3.1 provides the requirements for work package WP4 and WP5 on robot sensing and action, complementing and augmenting the detailed specification already provided in the work plan, as well as providing the architecture for the integration of the ROS nodes in each subsystem into a complete operational robot control system.

Contents

1 Introduction **4**

2 System Architecture Overview **5**

3 ROS Node Specifications **6**
3.1 Animate Behaviour 6
3.2 Behavior Controller 9
3.3 Face Detection 11
3.4 Gesture Execution 14
3.5 Overt Attention 18
3.6 Person Detection 22
3.7 Robot Localization 25
3.8 Robot Navigation 27
3.9 Sound Detection 29
3.10 Speech Event 31
3.11 Tablet Event 33
3.12 Text to Speech 35

4 System Architecture in Detail **37**

References **38**

Principal Contributors **39**

Document History **40**

1 Introduction

This deliverable, D3.1 System Architecture Design, provides a detailed specification of the architecture of the CSSR4Africa software system that controls the culturally sensitive social interaction between the Pepper robot and a human, i.e., a visitor in one of the two use case scenarios specified in Deliverable D2.1.

It specifies the architecture's component subsystems, the modules (i.e., ROS nodes) comprising each subsystem, and the information exchanged between each module. This includes a specification of the data that are input to each module, the data that are output from each module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed: through ROS topics, services, or actions, or through input and configuration files.

Based on the ROS nodes identified in Deliverables D2.2 and D2.3, Deliverable D3.1 provides the requirements for work package WP4 and WP5 on robot sensing and action, complementing and augmenting the detailed specification already provided in the work plan, as well as providing the architecture for the integration of the ROS nodes in each subsystem into a complete operational robot control system.

For each ROS node, the deliverable specifies the node configuration parameters that are read from the associated configuration file, the data that are read from the associated input file, the data that are written to an associated output file, the ROS topics to which the node subscribes for input, the ROS topics to which the node publishes output, the services that will be advertized and served, and the services that will be invoked.

The dynamics of the interaction between the visitor and the robot, implemented with these ROS nodes, will be specified in Deliverable D5.4.2 Robot Mission Language, and implemented in the `behaviorController` node in Deliverable D5.4.3 Robot Mission Interpreter.

In the following, we provide a high-level overview of the system architecture, a specification of each constituent ROS node, and detailed architecture that includes the topics published by each node, the topics each node subscribes to, and the services advertised and served by each node.

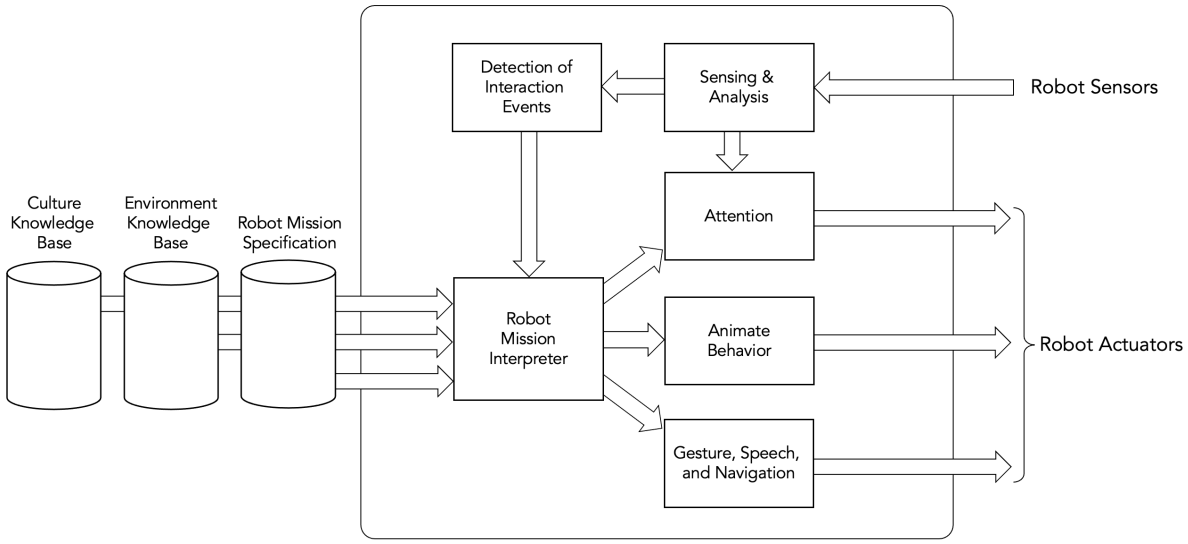

Figure 1: The CSSR4Africa system architecture.

2 System Architecture Overview

The CSSR4Africa system architecture comprises six subsystems, as follows; also see Figure 1.

1. Sensing and Analysis
2. Detection of Interaction Events
3. Animate Behaviour
4. Attention
5. Robot Mission Interpreter
6. Gesture, Speech, & Navigation

Each subsystem comprises one or more ROS nodes, as shown in Table 1, with thirteen ROS nodes in total. We provide a detailed specification of each ROS node in the following section. All thirteen nodes are part of the `cssr_system` ROS package; see Deliverable D3.2 Fig. 4 and Table 1.

Subsystem	ROS Node

Sensing and Analysis	`faceDetection`
	`personDetection`
	`soundDetection`

Detection of Interaction Events	`speechEvent` `tabletEvent`
Robot Mission Interpreter	`behaviorController`
Attention	`overtAttention`
Animate Behaviour	`animateBehaviour`

Gesture, Speech, & Navigation	`gestureExecution`
	`robotLocalization`
	`robotNavigation`
	`textToSpeech`

Table 1: The thirteen ROS nodes that comprise the six subsystems in the CSSR4Africa culturally-sensitive human-robot interaction software system.

3 ROS Node Specifications

3.1 Animate Behaviour

ROS Node Name

`animateBehaviour`

Functional Specification

This ROS node gives the robot the appearance of an animate agent by continually making subtle body movements, flexing its hands a little, and rotating its base slightly.

The node actuates the robot joints periodically in a random pattern, keeping the joint angles close to the default home positions. All the joints, except `headYaw` and `headPitch`, are actuated, as well as the wheels to effect rotation about the robot's z-axis, but not forward movement along the robot's x-axis. The `headYaw` and `headPitch` joints are actuated by the `attention` node. The extent of the movements is determined by an external parameter. Specifically, the range of movement, from which the actual movement will be a random sample, will be specified as a percentage of one tenth the full range of movement. Thus, 10% means that the amount of motion will vary randomly between the home value and half the maximum value, assuming the home value is midway between the minimum and maximum values.

The names of the topics to be used for each actuator are read from a data file comprising a sequence of key-value pairs. The key is the name of the actuator. The value is the topic name. There are two data files, one for the physical robot and another for the simulator.

To ensure that the robot does not make these animate movements when engaged in culturally sensitive social interaction through gestures, speech, or when navigating, the generation of animate behaviour is enabled and disabled by the Robot Mission Interpreter subsystem using a dedicated ROS service which the `animateBehaviour` node advertizes and serves.

Any of the three types of animate behaviour — body movement, hand flex, and rotation — can be selectively invoked. All three will be invoked if none are selectively invoked.

The node can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal.

Configuration File

The operation of the `animateBehaviour` node is determined by the contents of a configuration file, `animateBehaviourConfiguration.ini`, that contains a list of key-value pairs as shown below.

Key	Values	Effect
`platform`	`robot`, `simulator`	Specifies the platform on which the node is to be run, i.e., the physical Pepper robot or the Pepper simulator
`behaviour`	`body`, `hands`, `rotation`	Specifies the type of animate behaviour to exhibit.
`range`	`<number>`	Specifies the range of actuator movement as a percentage of half the full range of movement.
`robotTopics`	`pepperTopics.dat`	Specifies the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored.
`simulatorTopics`	`simulatorTopics.dat`	Specifies the filename of the file in which the simulator sensor and actuator topic names are stored.
`verboseMode`	`true`, `false`	Specifies whether diagnostic data is to be printed to the terminal.

Input Data File

This node does not read from an input data file.

Output Data File

This node does not write to an output data file.

Topics Subscribed

This node does not subscribe to any topics.

Topics Published

This node publishes on several actuator topics which are identified in the file given by the `robotTopics` and `simulatorTopics` key-value pairs in the configuration file. The following are the topics to which the `animateBehaviour` node publishes.

Topic	Actuator	Platform
/pepper_dcm/RightHand_controller/ follow_joint_trajectory	RHand	Physical robot
/pepper_dcm/LeftHand_controller/ follow_joint_trajectory	LHand	Physical robot
/pepper_dcm/RightArm_controller/ follow_joint_trajectory	RShoulderPitch, RShoulderRoll, RElbowYaw, RElbowRoll, RWristYaw	Physical robot
/pepper_dcm/LeftArm_controller/ follow_joint_trajectory	LShoulderPitch, LShoulderRoll, LElbowYaw, LElbowRoll, LWristYaw	Physical robot
/pepper_dcm/Pelvis_controller/ follow_joint_trajectory	HipRoll, HipPitch, KneePitch	Physical robot
/cmd_vel	WheelFL, WheelFR, WheelB	Physical robot
/pepper/RightArm_controller/ follow_joint_trajectory	RElbowYaw, RElbowRoll	Simulator
/pepper/LeftArm_controller/ follow_joint_trajectory	LElbowYaw, LElbowRoll	Simulator
/pepper/Pelvis_controller/ follow_joint_trajectory	HipRoll, HipPitch, KneePitch	Simulator
/pepper/cmd_vel	WheelFL, WheelFR, WheelB	Simulator

Services Supported

This node provides and advertizes a server for a service `/animateBehaviour/set_activation` to enable and disable the operation of the node, i.e., to activate or suspend the publishing of data on the actuator topics to give the appearance of an animate agent. It uses a package-specific msg, `State.msg` with just one field `string state`, with a value of either "`enabled`" or "`disabled`". Depending on the string value, `animateBehaviour` will be enabled or disabled. If the enable/disable request is successful, the service response is "1"; if it is unsuccessful, it is "0". The service is called by the `behaviorController` node, enabling or disabling animate behaviour, as needed.

The following summarizes the services supported.

Service	Message Value	Effect
/animateBehaviour/set_activation	enabled, disabled	Enable or disable animate behaviour

Services Called

This node does not call any services.

3.2 Behavior Controller

ROS Node Name

`behaviorController`

Functional Specification

This ROS node provides that interprets the robot mission language developed in Task 5.4.2 and implements the specification of the two use case scenarios in that language, as documented in Deliverable D5.4.2. In does so by recruiting the robot and visitor behaviors documented in Deliverables D2.2 and D2.3, and the cultural knowledge encapsulated in Deliverable D5.4.1, and realized in the robot sensing modules developed in Work Package 4 and the robot behavior modules developed in Work Package 5. The outcome of this task is described in Deliverable D5.4.3.

The dynamics of interaction between the robot and the visitor in the two use case scenarios are specified by robot mission specifications written in the robot mission language defined in Task 5.4.2. The `behaviorController` node interacts with other CSSR4Africa ROS nodes to realize these dynamics. It does this by calling various services. These are detailed below.

The node can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal.

Configuration File

The operation of the `behaviorController` node is determined by the contents of a configuration file, `behaviorControllerConfiguration.ini`, that contain a list of key-value pairs as shown below.

Key	Values	Effect
`scenarioSpecification`	`scenarioSpecification.dat`	Specifies the filename of the file in which the robot mission specificatgion is stored.
`verboseMode`	`true`, `false`	Specifies whether diagnostic data is to be printed to the terminal.

Input Data File

This node reads the interaction scenario robot mission specification file specified in the configuration file.

Output Data File

This node does not write to an output data file.

Topics Subscribed

This node subscribes to three topics, as follows.

Topic	Node	Platform
/speechEvent/text	speechEvent	Physical robot
/robotLocalization/pose	/robotLocalization	Physical robot
/personDetection/data	/personDetection	Physical robot

The /robotLocalization/pose and /personDetection/data topics are needed to allow navigation with respect to people in the robot's field of view.

Topics Published

This node does not publish to any topics.

Services Supported

This node does not provide and advertize any services.

Services Called

This node calls the following services to implement the interaction dynamics specified in the scenario robot mission specification.

Service	Message Value	Effect
/animateBehaviour/set_activation	To be defined	Enable and disable the operation of animateBehaviour
/gestureExecution/perform_gesture	To be defined	Initiate the performance of a required gesture
/overtAttention/set_mode	To be defined	Set the social, scanning, or location mode of attention
/robotNavigation/set_goal	<Point><Quaternion>	Navigate to a given location
/tabletEvent/prompt_and_get_response	To be defined	Seek input from the visitor
/textToSpeech/say_text	<string>	Say a text message

The type of the variables that are passed as an argument to the service calls has not yet been defined. This will be done when the node that services and advertizes these services are fully specified. Similarly, the type of the service call return value has not yet been defined. Again, this will be done when the node that services and advertizes these services are fully specified.

Helper Classes Used

This node uses an instantiation of a helper class CultureKnowledgeBase to read the culture knowledge base file and retrieve the required data using class access methods.

Similarly, it uses an instantiation of a helper class EnvironmentKnowledgeBase to read the environment knowledge base file and retrieve the required data using class access methods.

The type of the variables that are passed as arguments to the CultureKnowledgeBase helper class access methods and the EnvironmentKnowledgeBase helper class access methods have not yet been defined. These will be done when the helper classes are fully specified in Deliverables D5.4.1 and D5.4.2, respectively.

3.3 Face Detection

ROS Node Name

`faceDetection`

Functional Specification

This ROS node detects faces and their location in the field of view of the Pepper robot and and determine their gaze direction. It computes their position in an 2D image frame of reference. In addition, the region that the face occupies in the image is determined by computing the bounding box surrounding the face. If more than one face is present in the robot's field of view, then all of them are detected and localized.

To ensure coherence in detection and localization over time, each detected face is labelled (e.g., "Face 1") and the same label is assigned to that person in subsequent images. The label and the bounding box are be colour-coded, assigning different colours to different faces, and the same colour to a given face in each image in a sequence of images. If that face is no longer detected in one or more images (the number to be specified in a configuration parameter value) due to, for example, a false reject error, then that label is not reused. If that face reappears in a subsequent image, it is given a new label. As such, this module is only concerned with consistent detection of face over time, not recognition of previously detected people, and it is assumed that people don't change between images. Consequently, a person in one image is deemed to be the same one in a previous image if the spatial displacement of the person is less than a given tolerance (specified by a configuration parameter value).

Additionally, for each detected face, the ROS node assesses whether the person is gazing directly at the robot. This is accomplished using head pose estimation, which provides a simpler alternative to mutual gaze detection.

The node has two inputs: an RGB image and a depth image from either the robot's cameras or the Intel realsense camera.

The node has two outputs: an RGB image, with bounding boxes drawn around each detected face in both images, and an array of records, one record for each face detected.

The components of a record are the face label, the 2D image coordinates denoting the centroid of the bounding box, the distance (or depth) of the face represented by the centroid from the camera, and a Boolean value denoting whether mutual gaze is established between the corresponding face and Pepper robot.

The image is displayed in an openCV window when the node is operating in verbose mode (see below). The array of records is published to a topic named `/faceDetection/data`.

The names of the topics to be used for each actuator is read from a data file comprising a sequence of key-value pairs. The key is the name of the actuator. The value is the topic name.

The node can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal, and output images are also displayed in openCV windows.

Configuration File

The operation of the `faceDetection` node is determined by the contents of a configuration file, `faceDetectionConfiguration.json`, that contain a list of key-value pairs as shown below.

`camera`	`pepperCamera`, `realsense`	Specifies which RGB camera to use.
`algorithm`	`mediapipe`, `sixdrep`	Specifies which algorithm to use.
`mp_facedet_confidence`	`<number>`	Specifies the confidence threshold for the mediapipe face detection algorithm.
`mp_headpose_angle`	`<number>`	Specifies the maximum angular deviation (in degrees) for mediapipe head pose estimation.
`centroid_max_distance`	`<number>`	Specifies the maximum allowed distance (in pixels) between centroids for tracking continuity.
`centroid_max_disappeared`	`<number>`	Specifies the maximum number of frames a centroid can disappear before being considered lost.
`sixdrepnet_confidence`	`<number>`	Specifies the confidence threshold for the SixDRepNet pose estimation algorithm.
`sixdrepnet_headpose_angle`	`<number>`	Specifies the maximum angular deviation (in degrees) for SixDRepNet head pose estimation.
`sort_max_disappeared`	`<number>`	Specifies the maximum number of frames an object can disappear for Sort tracker before being removed.
`sort_min_hits`	`<number>`	Specifies the minimum number of consecutive hits required for Sort tracker initialization.
`sort_iou_threshold`	`<number>`	Specifies the Intersection over Union (IoU) threshold for Sort tracker associations.
`robotTopics`	`pepperTopics.dat`	Specifies the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored.
`verbose_mode`	`true`	Specifies whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows.

Input Data File

This node does not read from an input data file.

Output Data File

This node does not write to an output data file.

Topics Subscribed

This node subscribes to four topics: two RGB camera sensor topics (one published by the Pepper robot, one by the RealSense camera), and two depth camera topics (one published by the Pepper robot, one by the RealSense camera). These are specified in the files identified by the `robotTopics` key-value pair in the configuration file.

The following are the topics to which the `faceDetection` node subscribes.

Topic	Sensor / Node	Platform
`/naoqi_driver/camera/front/image_raw`	`FrontCamera`	Physical robot
`/naoqi_driver/camera/depth/image_raw`	`DepthCamera`	Physical robot
`/camera/color/image_raw`	`RGBRealSense`	Intel RealSense
`/camera/depth/image_rect_raw`	`DepthRealSense`	Intel RealSense

Topics Published

The following are the topics which the `faceDetection` node publishes.

Topic	Sensor / Actuator / Node	Platform
`/faceDetection/data`	`overtAttention`, `GUI`	Physical robot

Services Supported

This node does not support any services.

Services Called

This node does not call any services.

3.4 Gesture Execution

ROS Node Name

`gestureExecution`

Functional Specification

This ROS node provides the robot with the ability to execute five forms of gesture: deictic, symbolic, and iconic non-verbal hand gestures, and bowing and nodding body gestures.

The specifications for these gestures are in joint space, except for deictic gestures which are in Cartesian space. Some gestures, e.g., iconic and symbolic hand gestures, are specified by learning the required motions either by manual teleoperation, recording the joint angles, or by demonstration, using an RGB-D depth camera to determine the joint angles of human gestures in a skeletal model and mapping these to the robot joints. Other gestures, i.e., deictic hand gestures and body gestures, are specified by gesture parameters, such as the pointing location for deictic gestures and the degree of inclination for bowing and nodding, and the joint angles are computed using the kinematic model of the robot head, torso, and arms. For deictic gestures, which require the robot to point at objects in its environment and move its head to look at the object, the pose of the robot in the world frame of reference is also used.

If the arm cannot achieve the required pose for a deictic gesture, the robot rotates to make the pose achievable, returning to the original orientation once the gesture is complete. Futhermore, the arm returns to a neutral position by the robot's side when the gesture is complete.

Iconic and symbolic gestures are defined by descriptors that specify the final gesture joint configuration and the manner in which that configuration is achieved. Descriptors comprise four elements. Each element is a key-value pair, where the value can be an identifier, a number, a vector of numbers, or a vector of a vector of numbers.

The first key-value pair specifies the gesture type (e.g., `type iconic`, `type symbolic`).

The second key-value pair identifies the ID number (e.g., `ID 01`).

The third element defines the number of waypoints in the trajectory, including the start gesture joint configuration and the final gesture joint configuration.

The fourth element is a vector of joint angles vectors. The number of joint angle vectors is equal to the number of way points, including the start joint configuration and the final gesture configuration. Body gestures have three joints: knee pitch, hip pitch, hip roll. Iconic and symbolic gestures have five joints: shoulder pitch, shoulder roll, elbow yaw, elbow roll, and wrist yaw. Before beginning the gesture, the arm is moved from its current joint configuration to the start joint configuration, i.e., the joint angles specified in the first vector in the vector of vector of joint angles.

The number of elements in the vector of joint angles is determined by the gesture type.

Descriptors for each gesture are stored in an external descriptor file.

If an iconic or symbolic gesture involves two arms, they are treated as a composite of two individual gestures, one for each arm.

The joint angles for bow and nod body gestures, as well as hand deictic gestures, are computed at run time using the kinematic model of the robot and the bow angle, nod angle, or the location in the environment to which the robot should point. The bow angle, nod angle, and pointing location are provided as an input to the module, along with the time in milliseconds that should elapse between the start of the gesture and the end of the gesture.

The pointing location with respect to the robot body, specified by the shoulder pitch and shoulder roll angles, is computed from the pointing location in the world frame of reference (and supplied as an input to the module) and the pose of the robot in the world frame of reference (provided by the `robotLocalization` node. No waypoints are required for deictic gestures; the joints are actuated to achieve the target joint angles, interpolating linearly, or adjusting the joint angles, joint angular velocities, and joint accelerations to mimic biological movement by using a minimum jerk model of biological motion.

It is assumed that the knee pitch angle is fixed during a bow body gesture and that the bow angle corresponds to the change in the hip pitch angle with respect to the default hip pitch angle. Similarly, it is assumed that the

nod angle is the change in the head pitch angle with respect to the default head pitch angle. Finally, it is assumed that the arm and fingers are straight in a deictic gesture, with fixed values of elbow yaw, elbow roll, wrist yaw, and hand angles, so that the palm of the hand is directed upwards.

The input to the module is a record comprising the gesture type (e.g., `iconic`, `symbolic`, `deictic`, `bow`, `nod`), the gesture ID for symbolic or iconic gestures (e.g., `01`), the duration of the gesture in milliseconds, and either a bow angle in degrees (for a bow body gesture), or a nod angle in degrees (for a nod body gesture), or the three dimensional coordinates of a pointing location (for a deictic gesture). For deictic gestures, the module also inputs the current robot pose from the `robotLocalization` node.

The output is a sequence of joint angles, joint angular velocities, and, optionally, joint angular accelerations. Data is published on the appropriate topics, as required.

The names of the topics to be used for each actuator is read from a data file comprising a sequence of key-value pairs. The key is the name of the actuator. The value is the topic name.

The node can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal.

Configuration File

The operation of the `gestureExecution` node is determined by the contents of a configuration file, `gestureExecutionConfiguration.ini`, that contains a list of key-value pairs, as shown below.

Key	Values	Effect
`interpolation`	`linear`, `biological`	Specifies the interpolation type. This indicates how the joint angles that define the trajectory in joint space between the current joint angles and the gesture joint angles are computed for body gesture and hand deictic gestures, and between way points for iconic and symbolic gestures. The two options are: (a) independent linear interpolation of each joint angle, and (b) biological motion, selecting the sequence of joint angular velocities and joint accelerations to form a trajectory in time and joint space that mimics biological movement.
`gestureDescriptors`	`gestureDescriptors.dat`	Specifies the filename of the file in which the gesture descriptors are stored.
`robotTopics`	`pepperTopics.dat`	Specifies the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored.
`verboseMode`	`true`, `false`	Specifies whether diagnostic data is to be printed to the terminal.

Input Data File

This node does not read from an input data file.

Output Data File

This node does not write to an output data file.

Topics Subscribed

This node subscribes to one topic, published by `robotLocalization` node, which provides the pose of the robot.

The following are the topics to which the `gestureExecution` node subscribes.

Topic	Node	Platform
`/robotLocalization/pose`	`robotLocalization`	Physical robot

Topics Published

The following are the topics to which the `gestureExecution` node publishes. These are specified in the file identified by the `robotTopics` key-value pair in the configuration file.

Topic	Actuator	Platform
`/pepper_dcm/RightHand_controller/ follow_joint_trajectory`	`RHand`	Physical robot
`/pepper_dcm/LeftHand_controller/ follow_joint_trajectory`	`LHand`	Physical robot
`/pepper_dcm/RightArm_controller/ follow_joint_trajectory`	`RShoulderPitch`, `RShoulderRoll`, `RElbowYaw`, `RElbowRoll`, `RWristYaw`	Physical robot
`/pepper_dcm/LeftArm_controller/ follow_joint_trajectory`	`LShoulderPitch`, `LShoulderRoll`, `LElbowYaw`, `LElbowRoll`, `LWristYaw`	Physical robot
`/pepper_dcm/Pelvis_controller/ follow_joint_trajectory`	`HipRoll`, `HipPitch`, `KneePitch`	Physical robot
`/cmd_vel`	`WheelFL`, `WheelFR`, `WheelB`	Physical robot

Services Supported

This node provides and advertizes a server for a service `/gestureExecution/perform_gesture` to initiate the performance of a required gesture. It uses a package-specific msg, `Gesture.msg`. The message has several fields, as follows.

Field	Field Value	Field Type	Units
`gesture_type`	`iconic`, `symbolic`, `deictic`, `bow`, `nod`	String	
`gesture_id`	`<number>`	Integer	
`gesture_duration`	`<number>`	Integer	milliseconds
`bow_nod_angle`	`<number>`	Integer	degrees
`location_x`	`<number>`	Real	metres
`location_y`	`<number>`	Real	metres
`location_z`	`<number>`	Real	metres

If the `perform_gesture` request is successful, the service response is “1”; if it is unsuccessful, it is “0”. The service is called by the `behaviorController` node.

The following summarizes the services supported.

Service	Message Value	Effect
`/gestureExecution/perform_gesture`	`iconic`, `symbolic`, `deictic`, `bow`, `nod`	Perform an iconic, symbolic, or deictic gesture

Services Called

This node invokes the service `/overtAttention/set_mode` to move the robot's head to focus on the location required to gesture at.

Service	Message Value	Effect
`/overtAttention/set_mode`	`location` `x, y, z`	Move the head to focus on a location defined by the parameters

3.5 Overt Attention

ROS Node Name

`overtAttention`

Functional Specification

This ROS node provides the robot with the ability to direct the gaze of the robot to salient features in the environment or to a given location in the environment.

There are five modes of operation: a social mode, a scanning mode, a location mode, a seeking mode, and a disabled mode. The first is active when the robot is engaged in social interaction. In this mode, the salient features are people's bodies, faces, eyes, and voices. The second is active when the robot is not engaged in social interaction and it is scanning the environment looking for people with whom to interact. In this mode, the salient features also include objects of interest in the robot's environment, in addition faces. People's faces has a higher priority than other general features. In the second mode of operation, the robot switches its focus of attention after a short period of time and does not return directly to the original focus of attention, thereby scanning its environment. In the third mode of operation, the robot gazes at a given location in its environment. If the head cannot achieve the required pose to gaze at a given location, the robot rotates to make the pose achievable. In the fourth mode, seeking, the robot attempts to establish mutual gaze with a person nearby for a short duration, after which, if unsuccessful, it returns success or failure. To establish mutual gaze, the node searches for a face that is looking straight at the robot. The robot might have to rotate about its base as well as rotating its head when searching. In disabled mode, the head is centred and remains immobile. The mode of operation is selected on the basis of a service request (see below).

Two saliency maps are generated, one based on social features and one based on the combination of social and general conspicuous features. The first is based on the output of Task 4.2.2 Face & Mutual Gaze Detection and Localization, and Task 4.2.3 Sound Detection and Localization. The second is based on information-theoretic saliency and the output of Task 4.2.2 Face & Mutual Gaze Detection and Localization [1].

In the scanning attention mode, three processes are active. The first is a winner-take-all process to determine a single focus of attention from the candidates in the saliency map. This is effected by a selective tuning model [2, 3, 4]. The second is an Inhibition-of-Return (IOR) mechanism that attenuates the attention value of previous winning locations so that new regions become the focus of attention. The third is an habituation process to gradually reduce the salience of the current focus of attention, thereby ensuring that attention is fixated on a given point only for a limited period [5].

The robot's gaze is directed by publishing the appropriate messages on the topic that controls the `headYaw` and `headPitch` joints so that the gaze is centred on the focus of attention. This requires the calibration of the x and y offset of the focus of attention in the image to the change in `headYaw` and `headPitch` angles, respectively. In the case of aural attention to conspicuous sounds, it requires the calibration of the angle of arrival of the sound with the change in `headYaw angle`. Fixation on sounds will only control the `headYaw` angle, i.e., rotation in the horizontal plane about the head's z-axis.

These calibration constants are provided as parameters to the node. If the angle of rotation of the `headYaw` joint is greater than some threshold (defined as a parameter), then after rotating the head to fixate on the focus of attention, the base of the robot and the head rotate in opposite directions so that the robot continues to gaze at the focus of attention while it realigns its head with its body. The threshold for this head-torso realignment is provided as a parameter to the node.

The node has four inputs, all acquired by subscribing to the appropriate topics. These are the outputs of the `faceDetection` node and the `soundDetection` node to be used to compute the saliency map in social mode, an RGB image from the robot's top forehead camera or the additional RealSense external camera to be used to compute the saliency map in scanning mode, and, when attending to a given location in the environment, the current robot pose from the `robotLocalization` node.

The node has four outputs, effected by publishing to the topic to control the `headYaw` and `headPitch` joints, and, if required, the topic to control robot's wheels and the robot's angular velocity when adjusting its pose to recentre the gaze or enable the gaze to be directed at a given location. The third output is an RGB image depicting the saliency function and the selected focus of attention. The image is displayed in an openCV

window when the node is operating in verbose mode (see below). The fourth output is the current active mode indicating which mode of operation the node is in at that moment. The mode is continuously published to the `/overtAttention/mode` topic.

The names of the topics to be used for each actuator are read from a data file comprising a sequence of key-value pairs. The key is the name of the actuator. The value is the topic name.

To ensure that the robot does not rotate to realign its head and torso when navigating, attention is disabled by the `behaviorController` node using a dedicated service which the `overtAttention` node advertizes and serves. It can also enable attention by setting the mode to social, seeking, scanning, or location mode also effected by the `overtAttention` node using the same dedicated service which the `overtAttention` node advertizes and serves.

The node can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal, and output images are also displayed in openCV windows.

Configuration File

The operation of the `overtAttention` node is determined by the contents of a configuration file, `overtAttentionConfiguration.ini`, that contain a list of key-value pairs as shown below.

Key	Values	Effect
`camera`	`FrontCamera`, `RealSense`	Specifies which RGB camera to use.
`realignmentThreshold`	`<number>`	Specifies the threshold on the angular difference between head and base that must be met before the head and base are realigned.
`xOffsetToHeadYaw`	`<number>`	Specifies the calibration constant that defines the conversion of the offset in the (horizontal) x-axis of an image from the image center to the change in `headYaw` joint angle.
`yOffsetToHeadPitch`	`<number>`	Specifies the calibration constant that defines the conversion of the offset in the (vertical) y-axis of an image from the image center to the change in `headPitch` joint angle.
`robotTopics`	`pepperTopics.dat`	Specifies the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored.
`verboseMode`	`true`, `false`	Specifies whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows.

Input Data File

This node does not read from an input data file.

Output Data File

This node does not write to an output data file.

Topics Subscribed

This node subscribes to six topics: three from other nodes in the `cssr_system` package, and three camera sensor topics. These are are specified in the file identified by the `robotTopics` key-value pair in the configuration file.

The following are the topics to which the `overtAttention` node subscribes.

Topic	Sensor / Node	Platform
`/faceDetection/data`	`faceDetection`	Physical robot
`/robotLocalization/pose`	`robotLocalization`	Physical robot
`/soundDetection/direction`	`soundDetection`	Physical robot
`/naoqi_driver/camera/front/image_raw`	`FrontCamera`	Physical robot
`/camera/color/image_raw`	`RGBRealSense`	Intel RealSense
`/camera/depth/image_rect_raw`	`DepthRealSense`	Intel RealSense

Topics Published

This node controls five joints: `headYaw` and `headPitch` to adjust the head gaze, and `WheelFL`, `WheelFR`, and `WheelB` to rotate the robot and allow the gaze angle to be recentred. This node also publishes the current active mode of operation to a topic.

The specific topics that are used to control these joints are identified in the file given by the `robotTopics` key-value pair in the configuration file.

The following are the topics to which the `overtAttention` node publishes.

Topic	Actuator	Platform
`/pepper_dcm/Head_controller/follow_joint_trajectory`	`headYaw`, `headPitch`	Physical robot
`/cmd_vel follow_joint_trajectory`	`WheelFL`, `WheelFR`, `WheelB`	Physical robot
`/overtAttention/mode`		Physical robot

The following table details the custom message, Status.msg, that the attention node publishes on the `/overtAttention/mode` opic.

Field	Field Value	Field Type	Comment
`mode`	`social`, `scanning`, `location`, `seek`, `disabled`	String	
`value`	`1 (seeking), 2 (success), 3 (failure)`	Integer	

The value field only has meaning in seeking mode and scanning mode. The value 1 means the robot is attempting to establish mutual gaze. The value 2 means the robot has succeeded in detecting mutual gaze. And the value 3 means the robot failed to establish mutual gaze with anyone nearby. However, the value 3 has no meaning in scanning mode as it can keep scanning indefinitely because scanning mode is not time bound.

Services Supported

This node provides and advertizes a server for a service `/overtAttention/set_mode` to set the social, scanning, location, seeking, or disabled modes of attention. It uses a package-specific msg, `Mode.msg`. The message has four fields, as follows.

Field	Field Value	Field Type	Units
`state`	`social`, `scanning`, `location`, `seeking`, `disabled`	String	
`location_x`	`<number>`	Real	metres
`location_y`	`<number>`	Real	metres
`location_z`	`<number>`	Real	metres

If the request is successful, the service response is “1”; if it is unsuccessful, it is “0”. The service is called by the `behaviorController` node and the the `gestureExecution` node, setting the required mode of attention.

The following summarizes the services supported.

Service	Message Value	Effect
`/overtAttention/set_mode`	`social`, `scanning`, `location seeking`, `disabled`	Select mode of attention

Services Called

This node does not call any services.

3.6 Person Detection

ROS Node Name

`personDetection`

Functional Specification

This ROS node detects people in the field of view of the Pepper robot and determines their location. It computes their position in an 2D image frame of reference and in the robot's head 3D Cartesian frame of reference. In addition, the region that the person occupies in the image is determined by computing the bounding box surrounding the person. If more than one person is present in the robot's field of view, then all of them are detected and localized.

To ensure coherence in detection and localization over time, each detected person is labelled (e.g., "Person 1") and the same label is assigned to that person in subsequent images. The label and the bounding box are be colour-coded, assigning different colours to different people, and the same colour to a given person in each image in a sequence of images. If that person is no longer detected in one or more images (the number to be specified in a configuration parameter value) due to, for example, a false reject error, then that label is not reused. If that person reappears in a subsequent image, she or he is given a new label. As such, this module is only concerned with consistent detection of people over time, not recognition of previously detected people, and it is assumed that people don't change between images. Consequently, a person in one image is deemed to be the same one in a previous image if the spatial displacement of the person is less than a given tolerance (to specified by a configuration parameter value).

The node has two inputs: an RGB image from one of the robot's cameras and a depth image from one of the robot's depth sensors.

The node has three outputs: an RGB image and a depth image, with bounding boxes drawn around each detected person in both images, and an array of records, one record for each person detected.

The components of a record are the person label, the 2D image coordinates denoting the centroid of the bounding box, the width and height of the bounding box, a confidence value between 0 and 1 indicating the likelihood that the detection is not a false positive, and the 3D coordinates that define the point that corresponds to the centroid of the bounding box surrounding the person in the image.

The RGB image and the depth image are displayed in an openCV window when the node is operating in verbose mode (see below).

The array of records is published to a topic named `/personDetection/data`.

The names of the topics to be used for each sensor will be read from a data file comprising a sequence of key-value pairs. The key is the name of the sensor. The value is the topic name.

The node can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal, and output images are also displayed in openCV windows.

Configuration File

The operation of the `personDetection` node is determined by the contents of a configuration file, `personDetectionConfiguration.ini`, that contain a list of key-value pairs as shown below.

Key	Values	Effect
`algorithm`	`HOG`, `YOLO`	Specifies which algorithm to use.
`camera`	`FrontCamera`, `RealSense`	Specifies which RGB camera to use.
`falseRejectTolerance`	`<number>`	Specifies the number of images in which a person is not detected before it will be assigned a new label if and when it reappears at the same location. For example, a value 1 indicates that if the person is not detected in one image but reappears at that location in the next, it will be assigned the same label.
`spatialTolerance`	`<number>`	Specifies the spatial tolerance, given as a percentage of the width of the image, for a person to be assigned the same label.
`robotTopics`	`pepperTopics.dat`	Specifies the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored.
`verboseMode`	`true`, `false`	Specifies whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows.

Input Data File

This node does not read from an input data file.

Output Data File

This node does not write to an output data file.

Topics Subscribed

This node subscribes to five topics: four camera sensor topics and one depth sensor topic. These are are specified in the file identified by the `robotTopics` key-value pair in the configuration file.

The following are the topics to which the `personDetection` node subscribes.

Topic	Sensor	Platform
`/naoqi_driver/camera/front/image_raw`	`FrontCamera`	Physical robot
`/naoqi_driver/camera/depth/image_raw`	`DepthCamera`	Physical robot
`/camera/color/image_raw`	`RGBRealSense`	Intel RealSense
`/camera/depth/image_rect_raw`	`DepthRealSense`	Intel RealSense

Topics Published

The following are the topics to which the `personDetection` node publishes.

Topic	Sensor / Actuator / Node	Platform
`/personDetection/data`	`personDetection`	Physical robot

Services Supported

This node does not support any services.

Services Called

This node does not call any services.

3.7 Robot Localization

ROS Node Name

`robotLocalization`

Functional Specification

This ROS node determines the pose (position and orientation) of the robot in a Cartesian world frame of reference. It does this continuously, in real time, by updating the current pose based on relative pose estimation, using either odometry or the robot's inertial management unit IMU (or a combination of both). Since pose estimation errors using relative techniques grow with time, the module periodically resets its pose estimate using absolute pose estimation.

Absolute position estimation is accomplished by triangulation, using three landmarks (three are required because their distance from the robot is not known since the robot's range sensors are not sufficiently accurate). Landmark recognition is accomplished both using SIFT (Scale Invariant Feature Transform) and YOLO (You Look Only Once) real-time object detection. The position of the landmarks is extracted from a map of the environment. This map will be produced in Task 5.5.3 Environment Map Generation. The orientation of the robot is computed only for its rotation about the *Z*-axis; adjustments of body posture through rotation about the *X*- and *Y*-axes are ignored. This rotation angle is recovered by determining the direction given by the line of sight from the robot to one of the landmarks, and adjusting for any rotation about the *Z*-axis of the robot's head frame of reference with respect to the base frame of reference.

The node has five inputs: For relative pose estimation, the input is the odometry data published by the robot and data from the robot's accelerometer and gyrometer. For absolute pose estimation, the input takes the form of an RGB image from one of the robot's cameras. Input is also be acquired from the encoder on the head yaw actuator, i.e., the joint responsible for rotation in the azimuth (horizontal) plane. The names of the topics to be used for each sensor will be read from a data file comprising a sequence of key-value pairs. The key is the name of the sensor. The value is the topic name. The node also serves a `robotLocalization/reset_pose` service to reset the pose of the robot using absolute pose estimation. This service is typically called by the `robotNavigation` node.

The node has two outputs: an RGB image, with bounding boxes drawn around each detected landmark, and a record with the 2D pose information: x and y coordinates and rotation about the *Z*-axis. The image is displayed in an openCV window when the node is operating in verbose mode (see below). The record is published on a topic named `/robotLocalization/pose`.

The node can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal, and output images are also displayed in openCV windows.

Configuration File

The operation of the `robotLocalization` node is determined by the contents of a configuration file that contain a list of key-value pairs as shown below.
The configuration file is named `robotLocalizationConfiguration.ini`.

Key	Values	Effect
`camera`	`FrontCamera`, `RealSense`	Specifies which RGB camera to use.
`resetInterval`	`<number>`	Specifies the distance that can be travelled in centimetres before the relative pose estimate is reset using the absolute pose estimate.
`robotTopics`	`pepperTopics.dat`	Specifies the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored.
`verboseMode`	`true`, `false`	Specifies whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows.

Input Data File

This node does not read from an input data file.

Output Data File

This node does not write to an output data file.

Topics Subscribed

This node subscribes to six topics: two RGB camera sensor topics (one published by the Pepper robot, one by the RealSense camera), one depth camera topic (by the RealSense camera), an odometry topic, and inertial measurement unit (IMU) topic, and a joint states topic for the head yaw angle. These are specified in the files identified by the `robotTopics` key-value pair in the configuration file.

The following are the topics to which the `robotLocalization` node subscribes.

Topic	Sensor	Platform
`/naoqi_driver/camera/front/image_raw`	`FrontCamera`	Physical robot
`/camera/color/image_raw`	`RGBRealSense`	Intel RealSense
`/camera/depth/image_rect_raw`	`DepthRealSense`	Intel RealSense
`/naoqi_driver/odom`	`Odometry`	Physical robot
`/naoqi_driver/imu/base`	`IMU`	Physical robot
`/joint_states`	`Head Yaw`	Physical robot

Topics Published

The following are the topics to which the `robotLocalization` node publishes.

Topic	Sensor / Actuator / Node	Platform

`/robotLocalization/pose`	`gestureExecution`
	`overtAttention`
	`robotNavigation`
	`behaviorController`
		Physical robot

Services Supported

This node provides and advertizes a server for a service `/robotLocalization/reset_pose` to reset the pose of the robot using absolute pose estimation. It uses a generic msg, `Reset.msg` with just one field `string`, with a value "`reset`". If the reset request is successful, the service response is "1"; if it is unsuccessful, it is "0". The service is called by the `robotNavigation` node.

The following summarizes the services supported.

Service	Message Value	Effect
`/robotLocalization/reset_pose`	`reset`	Reset robot pose

Services Called

This node does not call any services.

3.8 Robot Navigation

ROS Node Name

`robotNavigation`

Functional Specification

This ROS node controls the locomotion of the Pepper robot so that it navigates its environment, in which there are fixed inanimate obstacles and moveable obstacles in the form of humans, from its current position along the shortest path to a destination position and orientation identified in the use case scenario robot mission specification. Navigation is effected by identifying waypoints along the navigation path and the robot moves from waypoint to waypoint.

The node optionally augments a metric workspace map of the robot's environment with obstacles corresponding to the location of any humans that have been detected in the robot's field of view by the `personDetection` node. The extent of the human obstacle is determined using culturally sensitive proxemics. This augmented workspace map is then used to generate a configuration space map that constrains the robot's path from its current location to its target location using either Dijkstra's algorithm and the A* algorithm. Waypoints are identified using one of two candidate techniques: equidistant waypoints and high path curvature waypoints. Locomotion from waypoint to waypoint is effected using one of two locomotion algorithms: Multiple Input Multiple Output (MIMO) and divide and conquer (DnQ).

The node has three inputs. The first is a record identifying the destination pose for the robot, specified by the x and y coordinates of the location and the direction the robot should face (i.e., the direction of the *X*-axis in the robot base frame, all specified in the workspace frame of reference). This record is part of a message in a service call by the `behaviorController` node.[1] The second is robot's current pose. This is provided by messages published to the `robotLocalization` node. The third input is the required cultural knowledge regarding proxemics. This is provided querying the African cultural knowledge base using an instantiation of a `CultureKnowledgeBase` helper class to read the culture knowledge base file and retrieve the required data using class access methods.

The node has two outputs. The first is sequence of forward velocity and angular velocity values published on the relevant `cmd_vel` topic. The full name of the `cmd_vel` topic is will be read from a data file comprising a sequence of key-value pairs. The second is the planned path drawn on a configuration space image. This image is displayed in an openCV window when the node is operating in verbose mode (see below).

The node can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal, and output images are also displayed in openCV windows.

Configuration File

The operation of the `robotNavigation` node is determined by the contents of a configuration file, `robotNavigationConfiguration.ini`, that contain a list of key-value pairs as shown below.

Key	Values	Effect
`map`	`scenarioOneMap.dat`	Specifies the filename of the file in which the workspace map is stored.
`pathPlanning`	`Dijkstra`, `A*`	Specifies the path planning algorithm to be used.
`socialDistance`	`true`, `false`	Specifies whether or not to take into consideration social constraints while navigating.
`robotTopics`	`pepperTopics.dat`	Specifies the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored.
`verboseMode`	`true`, `false`	Specifies whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows.

[1] A ROS action will replace the service if it is determined that feedback on achievement of the navigation goal is required.

Input Data File

This node does not read from an input data file.

Output Data File

This node does not write to an output data file.

Topics Subscribed

This node subscribes to one topic, as follows.

Topic	Node	Platform
`/robotLocalization/pose`	`robotLocalization`	Physical robot

Topics Published

The following are the topics to which the `robotNavigation` node publishes.

Topic	Sensor / Actuator / Node	Platform
`/cmd_vel`	`WheelFL`, `WheelFR`, `WheelB`	Physical robot

Services Supported

This node provides and advertizes a server for a service `/robotNavigation/set_goal` to request navigation to a given goal position and orientation. It uses a custom message to specify the pose with the x and y coordinates, and the angle of rotation θ about the z axis. If the navigation request is successful, the service response is "1"; if it is unsuccessful, it is "0". The service is called by the `behaviorController` node.

The following summarizes the service supported.

Service	Message Value	Effect
`/robotNavigation/set_goal`	`<x> <y> <theta>`	Define navigation goal pose

Services Called

This node calls the following two services.

Service	Message Value	Effect
`/robotLocalization/reset_pose`	`reset`	Reset the pose of the robot using absolute localization

Helper Classes Used

This node uses an instantiation of a helper class `CultureKnowledgeBase` to read the culture knowledge base file and retrieve the required data using class access methods.

Similarly, it uses an instantiation of a helper class `EnvironmentKnowledgeBase` to read the environment knowledge base file and retrieve the required data using class access methods.

The type of the variables that are passed as arguments to the `CultureKnowledgeBase` helper class access methods and the `EnvironmentKnowledgeBase` helper class access methods have not yet been defined. These will be done when the helper classes are fully specified in Deliverables D5.4.1 and D5.4.2, respectively.

3.9 Sound Detection

ROS Node Name

`soundDetection`

Functional Specification

This ROS node detects a conspicuous sound within the robot's hearing range and determines the direction of arrival of the sound.

Localization is limited to the azimuth (i.e., horizontal) plane. If a sound is detected, its direction of arrival will be determined in the robot's Cartesian head frame of reference based on the interaural time difference (ITD) between the arrival of the sound at the front left and front right microphones on the top of the robot's head. The node is tuned to detect human voices rather than ambient sounds or background noise by using signal processing techniques such as band-pass filtering.

The node has one inputs: the audio signal from the front left, front right, back left, and back right microphones, respectively. This data is published on `/naoqi_driver/audio`.

The node has two outputs: the angle of arrival in degrees relative to the robot head's forward-looking x-axis, and the audio signal of the detected sound captured by the front left microphone, from onset of the sound to offset.

The angle is published to a topic named `/soundDetection/direction`. The channels of the audio signal that is captured by the front left microphone is published to a topic named `/soundDetection/signal`. The final specification of this message type will be defined in Task 4.2.3 Sound Detection and Localization.

The names of the topics to be used for each sensor will be read from a data file comprising a sequence of key-value pairs. The key is the name of the sensor. The value is the topic name.

The node can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal.

Configuration File

The operation of the `soundDetection` node is determined by the contents of a configuration file, `soundDetectionConfiguration.ini`, that contain a list of key-value pairs as shown below.

Key	Values	Effect
`algorithm`	`ITD`	Specifies the localization technique to be used.
`lowFreqencyCutoff`	`<number>`	Specifies the low cutoff frequency in the band-pass filter in hertz.
`highFreqencyCutoff`	`<number>`	Specifies the low cutoff frequency in the band-pass filter in hertz.
`thresholdEnergy`	`<number>`	Specifies the threshold energy of the audio signal that qualifies it as a conspicuous sound.
`robotTopics`	`pepperTopics.dat`	Specifies the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored.
`verboseMode`	`true`, `false`	Specifies whether diagnostic data is to be printed to the terminal.

Input Data File

This node does not read from an input data file.

Output Data File

This node does not write to an output data file.

Topics Subscribed

This node subscribes to two microphone sensor topics. These are are specified in the file identified by the `robotTopics` key-value pairs in the configuration file.

The following are the topics to which the `soundDetection` node subscribes.

Topic	Sensor	Platform
`/naoqi_driver/audio`	`MicroFL_sensor`, `MicroFR_sensor` `MicroBL_sensor`, `MicroBR_sensor`	Physical robot

Topics Published

The following are the topics to which the `soundDetection` node publishes.

Topic	Sensor / Actuator / Node	Platform
`/soundDetection/direction`	`overtAttention`, `speechEvent`	Physical robot
`/soundDetection/signal`	`speechEvent`	Physical robot

Services Supported

This node does not support any services.

Services Called

This node does not call any services.

3.10 Speech Event

ROS Node Name

`speechEvent`

Functional Specification

This ROS node detects an utterance, represented as an audio signal, spoken by an interaction partner and transcribes it into written text.

The node uses a deep neural network that has been trained so that it can perform automated speech recognition. In the case that the spoken utterance cannot be recognized, either because the sound is not a spoken utterance or because it uses vocabulary on when the neural network has not been trained, then the node flags this by producing a text that reads "Error: speech not recognized".

The node has one input: an audio signal that is captured by the `soundDetection` module, published on a topic named `soundDetection/signal`.

The node has one output: a string representing the message in the spoken audio signal. This is published on a topic named `speechEvent/text`.

The node can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal.

Configuration File

The operation of the `speechEvent` node is determined by the contents of a configuration file, `speechEventConfiguration.ini`, that contain a list of key-value pairs as shown below.

Key	Values	Effect
`language`	`kinyarwanda`, `english`	Specifies the language in which the utterance is spoken.
`verboseMode`	`true`, `false`	Specifies whether diagnostic data is to be printed to the terminal.

Input Data File

This node does not read from an input data file.

Output Data File

This node does not write to an output data file.

Topics Subscribed

The following are the topics to which the `speechEvent` node subscribes.

Topic	Node	Platform
`soundDetection/signal`	`soundDetection`	Physical robot

Topics Published

The following are the topics to which the `speechEvent` node publishes.

Topic	Node	Platform
`/speechEvent/text`	`behaviorController`	Physical robot

Services Supported

This node does not support any services.

Services Called

This node does not call any services.

3.11 Tablet Event

ROS Node Name

`tabletEvent`

Functional Specification

This ROS node provides a ROS service to display a message, which will typically be a menu of interaction options, including a prompt requiring the selection of an option, wait for the visitor to select an option, and return the option selected. This service is called by the `behaviorController` node, in the Robot Mission Interpreter subsystem.

The node has one input: a number indexing the message to be displayed on the tablet PC, passed as an argument to a service call by the `behaviorController`.

The node has one output: a number representing option selected by the visitor, returned by the service call.

The node can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal.

Configuration File

The operation of the `tabletEvent` node is determined by the contents of a configuration file, `tabletEventConfiguration.ini`, that contains a list of key-value pairs as shown below.

Key	Values	Effect
`menuDataFile`	`tabletEventInput.dat`	Specifies the filename of the menu data.
`verboseMode`	`true`, `false`	Specifies whether diagnostic data is to be printed to the terminal.

Input Data File

This node reads the data file specified by the `menuDataFile` key-value pair.

Output Data File

This node does not write to an output data file.

Topics Subscribed

This node does not subscribe to a ROS topic.

Topics Published

This node does not publish to a ROS topic.

Services Supported

This node provides and advertizes a server for a service `/tabletEvent/prompt_and_get_response` to seek input from the visitor. It uses a package-specific msg, `Prompt.msg` with just one field `string message`. The value of the field is text to be printed on the tablet screen (the format specification is yet to be decided). The node then blocks for a predetermined time, waiting for the visitor to select an option. If the visitor responds within the specified time, the service response is option number $1 - n$; if they don't, the service response is zero. The service is called by the `behaviorController` node, setting the required mode of attention.

The following summarizes the services supported.

Service	Message Value	Effect
`/tabletEvent/prompt_and_get_response`	`<string>`	Print message, wait for input, and set response accordingly.

Services Called

This node does not call any services.

3.12 Text to Speech

ROS Node Name

`textToSpeech`

Functional Specification

This ROS node converts English, isiZulu, and Kinyarwanda text to speech. It uses a speech synthesis engine to convert text to an audio file that can then be played on the robots loudspeakers.

The node has one input: a string with the text to be spoken.

The node has one output: an audio signal representing the message in the spoken audio signal.

The node can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal.

Configuration File

The operation of the `textToSpeech` node is determined by the contents of a configuration file that contain a list of key-value pairs as shown below.
The configuration file is named `textToSpeechConfiguration.ini`.

Key	Values	Effect
`language`	`kinyarwanda`, `isiZulu`, `english`	Specifies the language in which the text is written.
`verboseMode`	`true`, `false`	Specifies whether diagnostic data is to be printed to the terminal.

Input Data File

This node does not read from an input data file.

Output Data File

This node does not write to an output data file.

Topics Subscribed

This node does not subscribe to any topics.

Topics Published

The following are the topics to which the `textToSpeech` node publishes. Note that the `/speech` topic accepts a text string, not an audio signal, and works best with the English language. Alternatives are being investigated.

Topic	Actuator	Platform
`/speech`	`LoudSpeakerLeft`, `LoudSpeakerRight`	Physical robot & Simulator

Services Supported

This node provides and advertizes a server for a service `/textToSpeech/say_text` to request the conversion of a text string to an audio signal, and to play that audio on the robot's loudspeakers.

It uses a standard message type, `std_msgs::String` defined in `std_msgs/String.msg`, to specify the text. If the request is successful, the service response is "1"; if it is unsuccessful, it is "0". The service is called by the `behaviorController` node.

The following summarizes the service supported.

Service	Message Value	Effect
`/textToSpeech/say_text`	`<string>`	Convert text to an audio signal and play it on the robot's loudspeakers

Services Called

This node does not call any services.

4 System Architecture in Detail

Figure 2 shows the CSSR4Africa system architecture specified using the ROS nodes, topics, and services specified in Section 3.

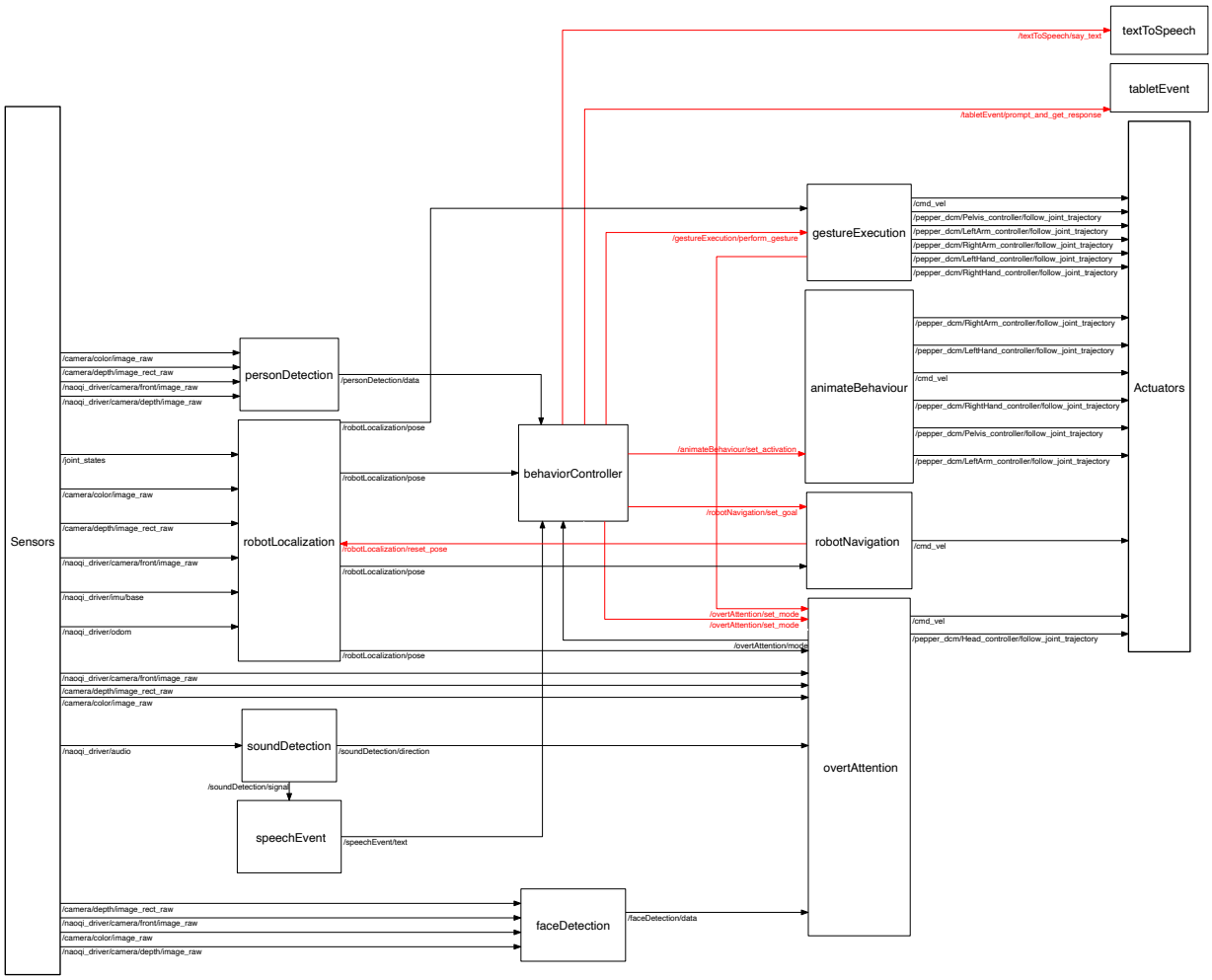

Figure 2: The CSSR4Africa system architecture specified at the level of ROS nodes (thin border rectangles), ROS topics (black lines), ROS services (red lines), and sources & sinks for sensor topics and actuator topics (thick border rectangles).

References

[1] N. D. B. Bruce and J. K. Tsotsos. Saliency, attention, and visual search: An information theoretic approach. *Journal of Vision*, 9(3):1–24, 2009.

[2] J. K. Tsotsos, S. Culhane, W. Wai, Y. Lai, N. David, and F. Nuflo. Modeling visual attention via selective tuning. *Artificial Intelligence*, 78:507–547, 1995.

[3] J. K. Tsotsos. Cognitive vision need attention to link sensing with recognition. In H. I. Christensen and H.-H. Nagel, editors, *Cognitive Vision Systems: Sampling the Spectrum of Approaches*, volume 3948 of *LNCS*, pages 25–36, Heidelberg, 2006. Springer.

[4] J. K. Tsotsos. *A Computational Perspective on Visual Attention*. MIT Press, Cambridge MA, 2011.

[5] A. Zaharescu, A. L. Rothenstein, and J. K. Tsotsos. Towards a biologically plausible active visual search model. In L. Paletta, J. K. Tsotsos, E. Rome, and G. Humphreys, editors, *Proceedings of the Second International Workshop on Attention and Performance in Computational Vision, WAPCV*, volume LNCS 3368, pages 133–147, Berlin, 2004. Springer.

Principal Contributors

The main authors of this deliverable are as follows (in alphabetical order).

Muhammed Danso, Carnegie Mellon University Africa.
David Vernon, Carnegie Mellon University Africa.
Yohannes Haile, Carnegie Mellon University Africa.

Document History

Version 1.0
First draft.
David Vernon.
24 January 2024.

Version 1.1
Added a configuration file for `tabletEvent`.
David Vernon.
25 January 2024.

Version 1.2
Added key-value pair for `personDetection` and `factDetection` to specify the algorithm to be used.
David Vernon.
01 February 2024.

Version 1.3
Updated topics for the `animateBehaviour` and `gestureExecution` nodes, replacing `command`, with `follow_joint_trajectory`. Updated topics for the `soundDetection` node.
Removed `/pepper/camera/stereo/image_raw` topic for simulator in the `faceDetection`, `overtAttention`, `personDetection`, and `robotLocalization` nodes.
Added `behaviorController` to the list of nodes to which `robotLocalization` publishes.
David Vernon.
09 May 2024.

Version 1.4
Updated topics for the `overtAttention` to include a topic on which the mode is published.
Muhammed Danso.
27 August 2024.

Version 1.5
Changed `scriptInterpreter` to `behaviorController`.
Amended some of the Version 1.4 updates.
David Vernon.
28 August 2024.

Version 1.6
Removed `waypointNumber` and `waypointSelection` keys from `robotNavigation`.
Changed the specification of the pose in the `robotNavigation/set_goal` service from using `geometry_msgs/Pose.msg` with `<Point> <Quaternion>` to a custom message with `<x> <y> <theta>` .
Changed references to script language and script interpreter to robot mission language and robot mission interpreter.
David Vernon.
09 September 2024.

Version 1.7
Updated scanning mode for the `overtAttention` to prioritize detected faces when scanning the environment for general features.
Updated `gestureExecution` to include head movement to complement pointing gestures. (with the help of Adedayo)
Updated `faceDetection` to remove eye detection and add mutual gaze detection. (with the help of Yohannes)
Updated Figure 2 system architecture. Muhammed Danso.
21 December 2024.

Version 1.8

Removed `/naoqi_driver/camera/stereo/image_raw` from `overtAttention`, `personDetection`, and `robotLocalization`. Added `/camera/color/image_raw` and `/camera/depth/image_rect_raw` to `overtAttention`, `personDetection`, and `robotLocalization`.
Changed option for selecting camera from `StereoCamera` to `RealSense` in `overtAttention`, `personDetection` and `robotLocalization`.
Updated the GraphViz architecture diagram to reflect these changes.
David Vernon.
1 January 2025.

Version 1.9

Changed the configuration file from `faceDetectionConfiguration.ini` to `faceDetectionConfiguration.json`
Updated the face Detection Configuration file key-value pairs table.
Fixed error on the Topics Subscribed overAttention node.
Removed a statement in the Sound Detection regrading message type.
Fixed error on Topics Published on Person Detection.
Blue text is used to mark addition of text or table entries and red text was used to mark deletion of text or table entries.
Yohannes Haile.
16 January 2025.

Version 1.10

Moved `behaviorController` to keep the ROS nodes in alphabetic order, and renamed the Section Robot Mission Interpreter to Behavior Controller.
David Vernon.
20 January 2025.

Version 1.11

Changed Interaction Scenario Manager subsystem to Robot Mission Interpreter subsystem.
Changed Interaction Manager in Figure 1 to Robot Mission Interpreter and added the Environment Knowledge Base.
Removed `knowledgeBase` node.
Added subsections on the `CultureKnowledgeBase` and `EnvironmentKnowledgeBase` helper classes in Sections 3.2 and 3.8, respectively.
David Vernon.
29 January 2025.

Version 2.0

Revised the system architecture to limit the support for the simulator. Simulator support will not be provided for ROS nodes that have the option of using an external device such as the RealSense camera or a LiDAR, or that use the Pepper microphones, speakers, or the tablet PC, or are dependent on data produced by these nodes. Consequently, the only node that supports the simulator is Animate Behavior. Since this is a major revision, the version number is advanced to 2.0.
David Vernon.
1 February 2025.

Version 2.1

Fixed incorrect reference to `personDetection` in documenting the topics to which the `robotNavigation` node publishes.
David Vernon.
9 February 2025.