CSSR for

Culturally Sensitive Social Robotics for Africa

D5.3 Attention Subsystem

Due date: **1/09/2024**
Submission Date: **06/12/2024**
Revision Date: **13/03/2025**

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

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

Responsible Person: **Muhammed Danso** Revision: **1.3**

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 document outlines the design and implementation of the Attention Subsystem for a culturally sensitive social robot being developed under the CSSR4Africa project. This deliverable represents the outcome of Task 5.3. Task 5.3 focuses on enabling the robot to dynamically direct its attention towards salient features in its environment, primarily during social interactions, and to scan its surroundings when not actively engaged. The Attention Subsystem integrates several processes, including face and sound detection, saliency map generation, and gaze control, allowing the robot to engage in more human-like behaviors. With modular design principles, the system incorporates different modes of operation such as disabled mode, social mode, scanning mode, seek mode, and location mode. This document details the system's requirements, functional specifications, and the modular design, offering a comprehensive view of the software architecture that powers the robot's attention capabilities. This subsystem enhances the robot's ability to interact naturally with its environment, contributing to its overall goal of promoting culturally appropriate human-robot interaction.

Contents

1 Introduction

The Attention Subsystem is a critical component of the CSSR4Africa project, aimed at developing a social robot that can interact effectively with people in a culturally sensitive manner. The subsystem's primary goal is to enable the robot to focus on relevant features in its environment, such as human faces and voices, or specific locations, while also scanning for points of interest when not actively engaged. This function is vital for the robot's ability to behave in an "animate" way, meaning that it can mimic human-like attention patterns by adjusting its focus based on its surroundings.

This document, Deliverable D5.3, provides a detailed description of the Attention Subsystem's design, including the software modules responsible for processing sensor data, managing attention modes, and controlling the robot's movements. The system is developed following a modular approach, with each function contributing to the overall behavior of the robot in various operational modes, such as social interactions or environment scanning. The architecture is designed to be flexible and adaptive, ensuring that the robot's attention system can be expanded or customized to accommodate different cultural contexts and interaction scenarios.

By integrating multiple sensory inputs and utilizing advanced saliency mapping techniques, the Attention Subsystem supports the robot's ability to maintain dynamic and contextually appropriate focus, improving both the usability and sociability of the robot. This deliverable elaborates on each functional module, providing insight into the system's implementation and future developments.

2 Requirements Definition

2.1 Overview

The objective of Task 5.3, Attention Subsystem, is to develop a software module that enables the robot to focus on salient features in its environment or a specified location. Salient features primarily include elements related to social interaction, such as people's bodies, faces, eyes, and voices. When the robot is not interacting, it scans the environment for interesting objects, contributing to the animate behavior of the robot (Task 5.2 Animate Behavior Subsystem). This scanning process requires a general-purpose saliency function to complement the social interaction feature saliency function.

2.2 Functional Specification

The Attention Subsystem operates as a ROS node and provides the robot with the ability to direct its gaze to salient features or specific locations in its environment. It has five modes of operation:

1. Social Mode: Active during social interactions, focusing on people's bodies, faces, eyes, and voices.
2. Scanning Mode: Active when the robot is not engaged in social interactions, scanning for people and interesting objects. In this mode, the robot's focus shifts periodically, avoiding returning to the initial focus point.
3. Location Mode: The robot gazes at a specified location. If the head cannot achieve the necessary pose, the robot rotates to align its head and body.
4. Seek Mode: In this mode the robot actively tries to establish mutual gaze with a person within close proximity for a fixed duration. During this time the robot either records success or returns a failure when the duration elapses.
5. Disabled Mode: Here the robot head is centred and remains immobile indefinitely.

2.3 Saliency Maps and Processes

Two types of saliency maps are generated:

- Social Features Saliency Map: Based on outputs from Task 4.2.2 Face & Mutual Gaze Detection and Localization, and Task 4.2.3 Sound Detection and Localization.
- General Features and Faces Saliency Map: Utilizing outputs from Task 4.2.2 and models such as the Itti and Koch saliency model [1, 2, 3], the model proposed by Rea et al. (2013) [4], the information-theoretic saliency map [5], and open-source deep neural network saliency models.

The subsystem incorporates three key processes in the scanning mode:

1. Winner-Take-All Process: Selects a single focus of attention from the saliency map candidates using a maximum function.
2. Inhibition-of-Return (IOR) Mechanism: Reduces the attention value of previous focus points to shift attention to new regions.
3. Habituation Process: Gradually reduces the salience of the current focus, limiting fixation duration.

2.4 Gaze Control

The robot's gaze is directed by publishing appropriate messages to control the headYaw and headPitch joints, centering the gaze on the focus of attention. Calibration of the x and y offsets of the focus of attention in the image correlates with changes in headYaw and headPitch angles, assuming a linear relationship. Using the image width, height, and camera field of view angles (horizontal and vertical) we compute the offset from the center and then calculate the proportional offsets relative to the image dimensions. Finally, we calculate and return the yaw and pitch angle changes. For specific formulas check equation (1) and (2) below. Aural attention calibration involves the angle of arrival of sound and headYaw angle adjustments. Fixation on sounds affects only the headYaw angle, which controls horizontal rotation about the head's Z-axis.

If the headYaw rotation exceeds a set threshold, the robot's base and head rotate in opposite directions to maintain focus while realigning the head with the body. Thresholds and calibration constants are provided as node parameters.

2.5 Inputs and Outputs

Topics Subscribed

This node subscribes to six topics: three from other nodes in the `cssr_system` package, one from naoqi, and two camera sensor topics. The specific camera sensor topic depends on whether the node is operating with the physical robot or the simulator. These are are specified in the files identified by the `robotTopics` and `simulatorTopics` key-value pairs 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/data`	`soundDetection`	Physical robot
`/joint_states`	`naoqi`	Physical robot
`/naoqi_driver/camera/front/image_raw`	`FrontCamera`	Physical robot
`/naoqi_driver/camera/stereo/image_raw`	`StereoCamera`	Physical robot

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 depend on whether the node is operating with the physical robot or the simulator. The corresponding topics 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 `overtAttention` node publishes.

Topic	Actuator	Platform
`/pepper_dcm/Head_controller/follow_joint_trajectory`	`headYaw`, `headPitch`	Physical robot
`/cmd_vel`	`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.

2.6 Data and Service Management

Topic names for actuators are read from data files containing key-value pairs. Separate files exist for the physical robot and the simulator. Attention mode settings are managed via a dedicated service advertised and served by the `overtAttention` node.

The node operates in normal or verbose mode, where verbose mode outputs published data to the terminal and displays output images in openCV windows. The `overtAttention` node's operation is defined by a configuration file (`overtAttentionConfiguration.ini`) containing key-value pairs, as shown below.

Key	Values	Effect
camera	FrontCamera, StereoCamera	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 re-aligned.
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.
robot Topics	pepperTopics.dat	Specifies the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored.
simulator Topics	simulatorTopics.dat	Specifies the filename of the file in which the simulator sensor and actuator topics are stored.
socialAttentionMode	saliency, random	Specifies whether to randomly choose a face or pick a face based on its saliency during social mode of operation.
verboseMode	true, false	Specifies whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows.

2.7 Implementation

The software development process encompasses requirements definition, module specification, interface design, module design, coding, and unit testing. The Attention Subsystem utilizes the outputs from Task 4.2.2 Face & Mutual Gaze Detection and Localization and Task 4.2.3 Sound Detection and Localization to achieve its objectives, ensuring robust social and environmental interaction capabilities for the robot.

3 System Design

Based on the requirements above we design a system architecture for the OvertAttention node. We follow a modular approach in this design by breaking the system into smaller functional units. The modules and services are as follows:

- Face detection callback
- Sound localization callback
- Camera sensor callback
- Robot localization callback
- Disabled mode function
- Social mode function
- Location mode function
- Scanning mode function
- Seek mode function
- Change Attention Function
- Attention application
- Mode service

3.1 Face detection callback

This function is set as a callback for subscribing to the `/facedetection/data` topic, which is published by the Face Detection Node. Its primary functionality is to receive coordinates of detected faces and compute the corresponding `HeadYaw` and `HeadPitch` angles. These angles are then stored in global variables for use by other modules in the system. The precise calculation ensures that the robot's gaze can be directed accurately towards the detected faces, facilitating effective social interaction.

3.2 Sound localization callback

This function is set as a callback for subscribing to the `/soundDetection/data` topic, published by the Sound Detection Node. Its sole purpose is to receive the direction of detected sounds and compute the corresponding `HeadYaw` angle. This angle is then stored in a global variable, allowing the robot to focus its attention on the source of the sound. By prioritizing aural attention, the robot can respond more naturally and effectively to auditory stimuli in its environment.

3.3 Camera sensor callback

This function is set as a callback for subscribing to topics such as `/naoqi_driver/camera/front/imageraw`, `/pepper/camera/front/imageraw`, or `/camera/color/image_raw`, depending on the configuration file. Its main functionality is to receive images from the robot's camera and store them in a global variable. These images are crucial for generating saliency maps and enabling the robot to visually scan its environment for interesting features.

3.4 Robot localization callback

This module is set as a callback for subscribing to the `/robotLocalization/pose` topic. It is responsible for receiving the current pose of the robot in a Cartesian world frame of reference and storing this information in a global variable. This data is essential for other functions that need to know the robot's pose to accurately adjust its head and body alignment, ensuring that the robot's movements and focus of attention are contextually appropriate and accurate.

3.5 Disabled mode function

This function centers the robot head and keeps it immobile and puts the `overtAttention` node to sleep for a short period of time. This mode is used to temporarily disable the attention functions of the robot, such as during navigation or when attention control is not required. It ensures that there's no conflict and the robot's resources are not unnecessarily utilized, maintaining system efficiency.

3.6 Social mode function

The Social Mode Function controls the robot's attention during social interactions. It relies on the detected faces and sound directions to determine where the robot should focus. A saliency map of face locations and sound locations is created first using outputs from the facedectection node and the sound localization node. The saliency of the faces is weighted by the corresponding distances to the robot and sound is given a default max weight. Then WTA is applied to select a focus of attention. Habituation and Inhibition Of Return (IOR) are applied to gradually reduce saliency and inhibit the selection of previous focus points respectively. Sound locations outside the camera field of view are not included in the saliency map, they are only directly attended to when there are no faces detected in the current FOV. The required `HeadYaw` and `HeadPitch` angle changes for the winning point in the saliency map is computed and used to call the Change Attention Function, directing the robot's gaze appropriately.

3.7 Location mode function

In Location Mode, the `OvertAttention` Node expects to receive location coordinates from the script interpreter or another node, focusing the robot's attention on these coordinates. This function uses the location information set by the mode service. It transforms the received x, y, and z coordinates into `HeadYaw` and `HeadPitch` angles and calls the Change Attention Function to adjust the robot's focus. This ensures the robot can accurately attend to specified locations when the functionality is needed.

3.8 Scanning mode function

The Scanning Mode Function is designed to focus the robot's attention on general interesting environmental features whilst prioritizing focus on faces of people. Using the global variable image set by the Camera Sensor Callback, it produces a saliency map and determines focal points. The function employs a selective tuning model to choose a focus point, an Inhibition-of-Return mechanism to shift attention from previous points, and a habituation process to limit fixation duration. The calculated `HeadYaw` and `HeadPitch` angles are then used to call the Change Attention Function, allowing the robot to scan and react to its surroundings dynamically.

3.9 Seek mode function

The seek mode function attempts to establish mutual gaze with a person within a fixed radius from the robot. This function rotates the robot about its base as well as rotating its head to make sure that all angles around the robot are covered. In each rotation the robot fixates for a fixed time to process the mutual gaze information from the face detection node and determine if mutual gaze has been established. Whenever mutual gaze is established the search stops and the robot stays in that pose, expecting a call to change `overtAttention` mode. If the robot covers the entire circular rotation about its base without establishing mutual gaze, it stops and remains immobile at its starting pose, again expecting a call to change `overtAttention` mode.

3.10 Change attention function

The Change Attention Function is pivotal in directing the robot's focus of attention by publishing precise control messages to adjust the `HeadYaw` and `HeadPitch` joints. When the computed `HeadYaw` rotation surpasses a predefined threshold, the function also publishes control messages to the robot's wheels, ensuring the alignment of the head and body. This realignment process utilizes the robot's current pose, provided by the Robot Localization Callback and stored in global variables. The function requires inputs including the rotation threshold and the desired `HeadYaw` and `HeadPitch` angles in radians. To achieve accurate adjustments, the function uses information from the `/joint_states` topic to continuously monitor the current positions of the `HeadYaw` and `HeadPitch` joints. By calculating the difference between the current and desired poses, the function sends appropriate commands to focus attention. During this process, the function temporarily puts the node to sleep, allowing the robot time to move. This ensures that the robot's attention remains precisely focused on the intended target, facilitating effective and natural interactions with its environment.

3.11 Attention application

The Attention Application is the main program that integrates all other functions and initializes the `OvertAttention` node. It starts by reading configuration parameters and initializing global variables. The application sets up the necessary publishers and subscribers, and determines the current mode and calls the corresponding function. The function for each mode completes only one process execution by calling the change attention function to publish the right message to the `HeadYaw` and `HeadPitch` joints and immediately returns back to the attention application. Therefore, each mode function stores it's state, if necessary, so that it can resume processing if that mode is still active. The current active mode is read from the global variable set by the mode service callback. The application runs in an infinite loop, continually checking the mode, publishing active mode information

on the `overtAttention/mode` topic, and processing callbacks to manage the robot's attention effectively.

3.12 Mode service

This advertises the `/overtAttention/set_mode` service, allowing the mode of attention to be set to social, scanning, seek, disabled, or location. 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, seek, disabled	String	
location_x	<number>	Real	metres
location_y	<number>	Real	metres
location_z	<number>	Real	metres

If the set mode request is successful, the service response is "1"; if it is unsuccessful, it is "0". The service is called by the `behaviorController` node, setting the required mode of attention. The callback of this service receives a Mode.msg entity and update global variables for mode and location_x, location_y, location_z coordinates.

4 System Implementation

The `overtAttention` node is implemented in C++ following the guidelines outlined in deliverable D3.2 Software Engineering Standards Manual using a modular design approach as detailed in the previous section.

The `overtAttention` node files in the `cssr_system` package are structured as follows:

```
cssr_system
└── overtAttention
    ├── config
    │   └── overtAttentionConfiguration.ini
    ├── data
    │   ├── pepperTopics.dat
    │   └── simulatorTopics.dat
    ├── include
    │   └── overtAttention
    │       └── overtAttentionInterface.h
    ├── msg
    │   ├── Mode.msg
    │   └── Status.msg
    ├── src
    │   ├── overtAttentionApplication.cpp
    │   └── overtAttentionImplementation.cpp
    ├── srv
    │   └── setMode.srv
    ├── README.md
    └── CMakeLists.txt
```

Figure 1: File structure for overtAttention node

The `overtAttentionConfiguration.ini` file contains the configuration parameters that control the behavior of the system as mention in earlier sections. Similarly, the `pepperTopics.dat` file contains the robot topics and the `simulatorTopics.dat` file contains the simulation environment topics. The `overtAttentionInterface.h` file contains all includes for imported libraries and function and global variable declarations. `Mode.msg` file defines a custom message type for service call to switch the mode of operation of the node. `Status.msg` file defines a custom message for the data type published to keep track of the status of current operating mode.

Furthermore, the `overtAttentionImplementation.cpp` file contains the definition of all functions used in achieving the objectives of this system. The `overtAttentionApplication.cpp` file has the main method of the program and it defines the step by step operation of the `overtAttention` node. It contains calls to the functions defined in `overtAttentionImplementation.cpp` and ties all of them together to achieve the aims of the system.

The `setMode.srv` file contains the definition of the `/overtAttention/set_mode` service advertised by the `overtAttention` node.

4.1 Change attention

A central function of the `overtAttention` node is to control the robot's head movements, ensuring it transitions logically and smoothly from one pose to another. At the heart of this implementation lies the function `move_robot_head_biological_motion`, which takes the head pitch and yaw as input parameters to position the head accordingly. This function leverages interpolation techniques to create smooth, fluid head movements that mimic natural, biological motion, avoiding the stiffness or jerkiness often associated with robotic movements.

4.2 Attention application

The main function of the `overtAttention` application begins by reading the configuration file to initialize its settings. It then advertises the `/overtAttention/set_mode` service and the `/overtAttention/mode` topic, followed by subscribing to `/joint_states`, `/soundDetection/direction`, `/faceDetection/data`, and the relevant camera topics. Once set up, the program enters a continuous loop, running as long as the ROS process remains active. Within this loop, a switch statement monitors the global mode variable, which is updated by the callback of the `/overtAttention/set_mode` service. Based on the current value of this variable, the appropriate mode function is invoked to handle operations accordingly.

4.3 Scanning Mode

In the implementation of the scanning mode operation, we employ a saliency detection method proposed by Xiaodi Hou and Liqing Zhang, based on a spectral residual approach [6]. Outputs from Task 4.2.2 are used to give locations of detected faces a slightly higher saliency value on the saliency map generated from the spectral residual approach. This allows the algorithm to put more emphasis on the faces of people in the robot's environment. To ensure the robot does not repeatedly focus on the same locations, we implement a habituation mechanism: the saliency value of previously selected locations are reduced by 0.1 per frame over 50 frames. After habituation, the location is blocked by assigning it a saliency value of 0 for the subsequent 50 frames, ensuring it is not selected again during this time.

The scanning mode operation follows a structured sequence of steps: saliency detection, application of habituation, inhibition of return (IOR), and identification of the most salient location using a max function. Finally, the robot's head is directed to focus on this winning location.

To support effective habituation and IOR, we account for the robot's head movements by mapping image coordinates to angular changes relative to the robot's base pose. This mapping ensures accurate head positioning. The calculations are as follows:

$$\delta Y = (x - (W/2))/W * \theta h \tag{1}$$

$$\delta P = (y - (H/2))/H * \theta v \tag{2}$$

Here, δY and δP represent the required changes in head yaw and pitch, respectively, while x and y are the pixel coordinates in the image. H and W denote the image's height and width, and θv and θh represent the camera's vertical and horizontal fields of view. The center of the camera (mounted in the robot's head) serves as the reference point for these calculations, with adjustments made to the pitch angle to accurately align the robot eyes with the desired pixel location.

The angle changes δY and δP do not directly represent unique angles in the real world because they are relative to the current pose of the robot's head, rather than its base. To obtain the absolute

angles in the real-world frame, we add the current yaw and pitch joint values (recorded at the time the image is captured) to δY and δP, respectively. This calculation provides the actual angles corresponding to the point—or more precisely, the line in the real-world environment since we are dealing with 2D images.

These updated yaw and pitch angles are then stored and later transformed back into pixel coordinates by reversing the earlier formulas and steps when applying habituation and inhibition of return (IOR) to the saliency map. This process ensures consistent and accurate mapping between image pixels and real-world coordinates during saliency-based operations.

4.4 Location Mode

The implementation of location mode is straightforward. The given x, y, z world coordinates are transformed into head pitch and head yaw angles using inverse kinematics. The head is then focused on the location using these angles.

4.5 Social Mode

In social mode we tried two implementation strategies and we found both useful after numerous tests so we decided to keep them and give users the choice to decide which one to use by changing the value of `socialAttentionMode` key in the configuration file. The two valid values are `saliency` or `random`. As mentioned in social mode we make use of faces and voices information. And with these two streams of information there are four cases namely:

- Case 1: when there is face signal and voice/sound signal
- Case 2: when there is face signal but no voice/sound signal
- Case 3: when there is no face signal but there is voice/sound signal
- Case 4: when there is no face signal and no voice/sound signal

In Case 1, when using `random` strategy, a face is chosen randomly from the list of faces and the head is focused on this face, unless the number of frames that have passed since the node started running is a multiple of 20, in which case attention is focused on the direction of the voice/sound. When using the `saliency` strategy, the faces are projected onto a saliency map with the saliency for each face relative to its distance from the robot where closer faces are more salient. The voice direction is also projected to the map if it is within the horizontal field of view (HFOV) of the camera, that means all voices coming from an angle outside the HFOV are ignored. Next the projected faces and voice direction are habituated, that is if a projected face or voice lands at a location which was previously a winner recently (we take recently in this case to mean within 3 frames) then its saliency is reduced. Afterwards, the most salient location on the saliency map is picked and the head is focused on the corresponding face or voice direction.

In Case 2, when using `random` strategy, there is no voice direction information so a face is chosen randomly from the list of faces and the head is focused on this face. When using the `saliency` strategy, the faces are projected onto a saliency map as above. And the most salient location on the saliency map is picked and the head is focused there.

In Case 3, using either of the strategies results in the same thing. The head is focused towards the direction of sound.

In Case 4, regardless of the strategy there is nothing to do as there is no information signal so the robot head stays immobile.

4.6 Seeking Mode

In the implementation of seeking for mutual attention we defined a fixed list of angles to turn the head to with respect to the pose of robot base. The list is as follows: $[-40, 0, 40, 0]$. We loop through this list to move the head to this yaw angles whilst keeping the pitch angle constant at 0. If mutual gaze is established at any of these angles then the head stays focused there, until `/overtAttention/set_mode` service is called again. When at the end of the list and mutual gaze is not established then the robot rotates 90 degrees about its base and the program loops through the angles again from the beginning. The robot rotates about its base for a maximum of four times in a single seeking mission, after which the robot would be facing where it started from and at this point it means mutual gaze couldn't be established, thus seeking ends and the robot remains immobile until `/overtAttention/set_mode` service is called.

4.7 Disabled Mode

Our implementation for the disabled mode is simple. The only step is to call `move_robot_head_biological_motion` to focus the head at the center and look forward.

5 Running Attention Node

The following commands will launch the `overtAttention` node (after waking up the robot):

```
# Launch the overtAttention node
rosrun cssr_system overtAttention
```

When the node is launched before it operates fully it will wait for the `/joint_states`, `/faceDetection/data`, `/soundDetection/direction`, and `robotLocalization/pose` topics to be available. The status of subscribing to these topics will be printed on the terminal accordingly. When these are successful the node will run normally and regularly print a heartbeat message to the terminal to let the user know it is running fine. These topics can be made available by one of two ways, outlined below:

Option 1: Running the `faceDetetction`, `soundDetection`, and `robotLocalization` nodes in the `cssr_system` package (if available)

```
# Launch the faceDetection node
rosrun cssr_system face_detetection_application.py
```

```
# Launch the soundDetection node
rosrun cssr_system sound_detection_application.py
```

```
# Launch the robotLocalization node
rosrun cssr_system robotLocalization
```

Option 2: Running the `overtAttentionDriver` in the `unit_tests` package and providing the actual robot pose in the physical environment.

```
# Launch the overtAttentionDriver node
rosrun unit_tests overtAttentionDriver <robot_x> <robot_y>
<robot_theta>
```

6 Unit Tests

Testing is an important step in the software development process as it helps us to validate the expected behavior of software, identify and fix bugs early. In this system we employed the unit testing mechanism as stipulated in Deliverable D3.5, System Integration and Quality Assurance.

The `overtAttention` node has five distinct modes of operation namely: disabled mode, location mode, social mode, scanning mode, and seeking mode. Switching between these modes can only be done through service calls to the `/overtAttention/set_mode` service advertised by the `overtAttention` node. Each of these modes of operation have a combination of functions and code isolated from the rest. Therefore, in the unit test we test the five modes separately as independent units.

The `overtAttention` node uses the outputs from Task 4.2.2 Face & Mutual Gaze Detection and Localization and Task 4.2.3 Sound Detection and Localization to perform its operations. But the test needs to be for an independent and isolated `overtAttention` node. Thus, we created two drivers to generate and publish data in place of outputs from these two tasks. The data generation for each driver is random and follows the same message structure as expected from outputs of the two tasks.

To implement the unit tests we made use of the GoogleTest library. The unit test for each mode is isolated in one function. Each test function starts by making a service call to `overtAttention` node to switch to the target mode for that function and retrieving the response message. The next step determines if the response message indicates a successful switch of mode, if not the test fails. If successful the function calls the sleep function to allow for the `overtAttention` node to operate in this mode for a few seconds. The next step is to log the test's success or failure to an output file and end the test.

Additionally, the test function for seeking mode does a little bit more. After a successful call to switch the `overtAttention` node's mode to seeking, it waits for the `overtAttention` node to publish that seeking is complete by listening to messages published on the `overtAttention/mode` topic. Then it parses this message to determine if seeking was successful or it failed to establish mutual gaze. It then compares this to what is expected based on the data published by the drivers. If they do not match then the test fails. The remaining steps are the same as explained above for other test functions.

The file structure for the `overtAttention` node in the unit tests package is as follows:

```
unit_tests
└── overtAttentionTest
    ├── config
    │   └── overtAttentionTestConfiguration.ini
    ├── data
    │   ├── overtAttentionTestOutput.dat
    │   ├── pepperTopics.dat
    │   └── simulatorTopics.dat
    ├── include
    │   └── overtAttentionTest
    │       └── overtAttentionTestInterface.h
    ├── launch
    │   ├── overtAttentionLaunchRobot.launch
    │   ├── overtAttentionLaunchSimulator.launch
    │   └── overtAttentionLaunchTestHarness.launch
    ├── msg
    │   └── faceDetection.msg
    ├── src
    │   ├── overtAttentionTestApplication.cpp
    │   ├── overtAttentionTestDriver.cpp
    │   └── overtAttentionTestImplementation.cpp
    ├── README.md
    └── CMakeLists.txt
```

Figure 2: File structure for overAttention unit test

The overtAttentionTestConfiguration.ini file contains key-value pairs per line. The keys are: scanning, social, seeking, location, and disabled. The keys take a value of either `true` or `false` indicating which modes should be tested when the program is run.

The overtAttentionTestOutput.dat is the output file where the test results are logged for future reference. The file is erased at the beginning of a test run. Thus, the file only contains log details of the last test run at any instant.

The launch files are used to launch the unit tests for `overtAttention`. `overtAttentionLaunchTestHarness.launch` launches the `overtAttention` node, unit test application, and driver nodes.

The `overtAttentionTestLaunchRobot.launch` file accepts the following parameters:

- `robot_ip`: specifies the IP address of the robot.
- `roscore_ip`: specifies the IP address of the roscore.
- `network_interface`: specifies the network interface name.

The following commands will launch the robot, start the `overtAttention` node, start the `overtAttentionTestDriver` node, and start the `overtAttentionTestApplication` to perform the tests:

```
# Launch the physical robot
roslaunch unit_tests overtAttentionLaunchRobot.launch \
robot_ip:=<robot_ip> roscore_ip:=<roscore_ip> \
network_interface:=<network_interface_name>

# Launch all the nodes
roslaunch unit_tests overtAttentionLaunchTestHarness.launch \
launch_drivers:=true launch_test:=true initial_robot_x:=<robot_x> \
initial_robot_y:=<robot_y> initial_robot_theta:=<robot_theta>
```

NOTE: Setting the argument `launch_ test` to `true` runs the tests based on the configuration, while setting it to `false` only launches the node and its dependencies. This is particularly useful if the user wants to run unique tests on the node. Setting the argument `launch_ drivers` to `true` launches the drivers and stubs required to drive the `overtAttention` node from the `unit_tests` package, while setting it to `false` launches the actual nodes (if available) which include `faceDetection`, `soundDetection`, and `robotLocalization` from the `cssr_system` package. The default values are `true`. Ensure that the robot pose matches the exact pose of the robot in the world frame of reference.

References

[1] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1254ñ–1259, 1998.

[2] L. Itti and C. Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 40:1489–1506, 2000.

[3] L. Itti and C. Koch. Computational modelling of visual attention. *Nature Reviews Neuroscience*, 2:194–203, 2001.

[4] F. Rea, G. Metta, and C. Bartolozzi. Event-driven visual attention for the humanoid robot iCub. *Frontiers in Neuroscience*, 7(Article 234):1–11, December 2013.

[5] 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.

[6] Xiaodi Hou and Liqing Zhang. Saliency detection: A spectral residual approach. In *2007 IEEE Conference on computer vision and pattern recognition*, pages 1–8. Ieee, 2007.

Principal Contributors

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

Adedayo Akinade, Carnegie Mellon University Africa.
Muhammed Danso, Carnegie Mellon University Africa.
David Vernon, Carnegie Mellon University Africa.

Document History

Version 1.0

First draft.
Muhammed Danso.
06 December 2024.

Version 1.1

Revised draft and made changes to scanning mode to prioritize faces detected whilst scanning the environment.
Muhammed Danso.
20 December 2024.

Version 1.2

Removed all references to simulator platform and provided additional information on running the unit tests.
Adedayo Akinade.
05 February 2025.

Version 1.3

Changing tenses in the requirement specification to present tense and removing assigned persons.
Muhammed Danso.
13 March 2025.