Demo-maker Quick-start Guide

Bastien Mourgue

May 2019

Contents

1 Introduction

The intent of this guide is to quickly present the essential points for a live demonstration of our algorithms using the demo-maker graphical user interface.

The demo-maker is a software that allows user to create different representations of the results, to configure and start the different algorithms required for the live demonstration. The word representation means all the panels which display the results of the different algorithms. In this guide, we will use a predefined set of representation. The creation of a new representation is not explained. All the algorithms and the demo-maker use ROS middle-ware to communicate between each other. Some basic knowledge on ROS are recommended.

2 Configuration

Before using ROS, the middle-ware should be initialized and the server should be started. Then, the demo-maker need some environment variables to be able to communicate with the middle-ware, and to configure correctly the path required for the software.

2.1 Ros Server

The first thing to do is to start the ROS server, called roscore. For that, open a new terminal and enter the command roscore. If everything goes well, you should have something similar than:

```
$ roscore

Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://<machine_name>:33919/
ros_comm version 1.4.7

SUMMARY
========
PARAMETERS
```

```
 * /rosversion
 * /rosdistro

NODES

auto-starting new master
process[master]: started with pid [13054]
ROS_MASTER_URI=http://<machine_name>:11311/

setting /run_id to 9cf88ce4-b14d-11df-8a75-00251148e8cf
process[rosout-1]: started with pid [13067]
started core service [/rosout]
```

The roscore can not boot if the network is badly configured (this is the case when the DNS could not resolve autimatically the IP address of the computer). In this case, you have to set the environment variable `ROS_IP` to specify the IP address. This address could be found using the command `ifconfig`.

```
$ export ROS_IP=xxx.xxx.xxx.xxx
$ roscore
```

One important thing to note in the output of the roscore command is the `ROS_MASTER_URI`. It contains the address and port for communication with master.

Note: The environment variables `ROS_IP and ROS_MASTER_URI` should be set in all terminals used for the demonstration.

2.2 Demo-maker environment variables

Open a new terminal and go to litbot directory. The first thing to do is to source the ros environment of the litbot package. For that, please enter the following command:

```
$ cd /path/to/folder/litbot/
$ source ros_workspace/devel/setup.[bash, sh, zsh]
```

The extension of the file relies on the `Shell` you are using.

The next step is to configure the ROS environment in order to be able to contact the ROS server :

```
$ export ROS_IP=xxx.xxx.xxx.xxx
$ export ROS_MASTER_URI=http://<machine-name>:11311
```

If the roscore and the demo-maker GUI are running in the same computer and the hostname could be resolved by the DNS, it is not mandatory to set these variables.

The demo-maker GUI also requires to position some environment variables.

- The folder where the plugins library are stored.

  ```
  $ export DEMOMAKER_PLUGINS_PATH=[litbot_dir]/
      ↪ ros_workspace/build/litbot/visualization/qt_visu
      ↪ /plugins
  ```

  If you are using the installed version, use instead:

  ```
  $ export DEMOMAKER_PLUGINS_PATH=[install_dir]/
      ↪ demo_maker/plugins
  ```

- The root folder of the demo-maker

  ```
  $ export DEMOMAKER_PLUGINS_PATH=[litbot_dir]/
      ↪ global_workspace/visualization/qt_visu/
  ```

  If you are using the installed version, use instead:

  ```
  $ export DEMOMAKER_PLUGINS_PATH=[install_dir]/
      ↪ demo_maker/
  ```

3 Start demo-maker GUI

When you are here, the environment should be configured.

3.1 Open a demo in the demo-maker GUI

To start the demo-maker GUI, the base command is:

```
$ rosrun litbot demo-maker
```

If the environment is well configured, then you should have the following screen (see Figure 1). Otherwise, something is missing in the configuration. For example, if the plugins path is not correctly set, nothing will be displayed in the bottom left widget (no nodes).

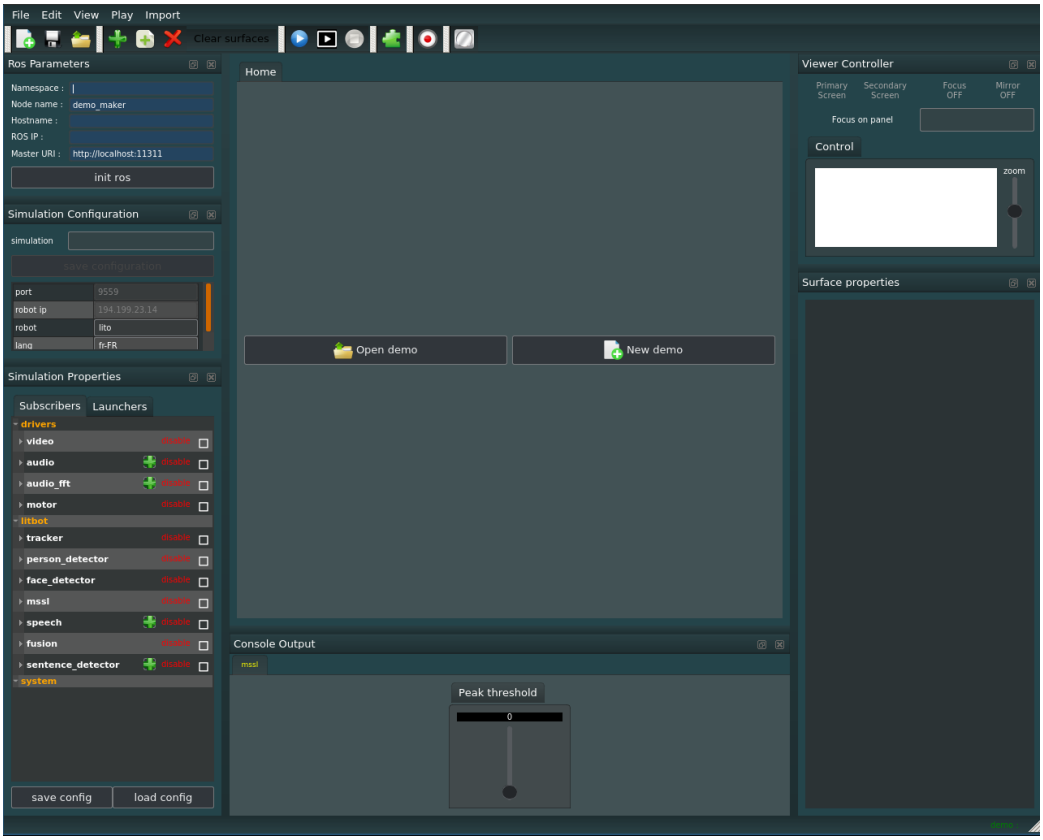

Figure 1: Home page of the demo-maker when the environment is well configured.

The first visible screen is the home page of the demo-maker. You can open an existing demonstration file or create a new one. Click on the `Open Demo` button and select the demo file. This file is located at:

```
$ [litbot_dir]/global_workspace/demo_config/litbot_new_debug.
    ↪ demo
```

or if you use the installed version

```
$ [install_dir]/share/demo_config/litbot_new_debug.demo
```

You can also open a demo using the key shortcut `Ctrl-o`. Once the demo file is loaded, you should have the following screen (Figure 2):

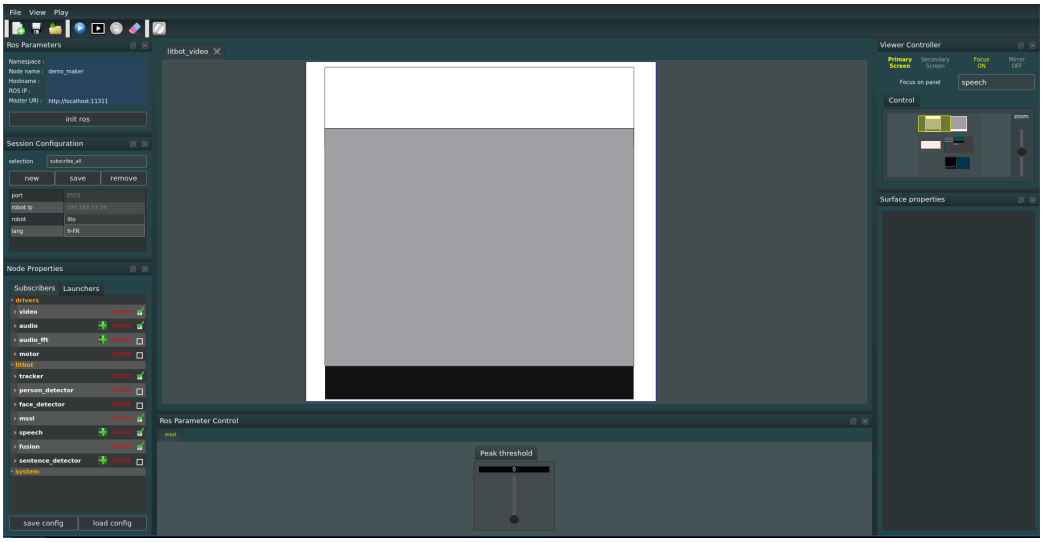

Figure 2: Demo-maker GUI when the demo litbot_new_debug.demo is loaded.

3.2 The command line options

Some options could be added to the command line in order to automatically configure the interface.

-d	--enable-designer		"Enable designer features"
-df	--demo-file	param_value	"Load the demo configuration file when opening the application"
-h	--help		"Display the current message"
-i	--ros-auto-init		"Initialize ros when opening the application"
-nc	--node-configuration	param_value	"Load the node configuration file (.config) when opening the application"
-ns	--namespace	param_value	"Global namespace used for all the node"
-rec	--enable-recorder		"Enable recording features"

The `-d` and `-rec` options should not be used during a live demonstration (features are not useful). You can directly open the demo file using the option `-df`, load the launcher configuration file using the `-nc` option, automatically initialize ros using the option `-i` and set a global namespace with the option `ns`.

All of these options are used to save some time, but everything could be configured directly in the interface.

The last step is to initialize ROS . This is done in the `Ros Parameters` (See Figure 3) widget of the interface. The field are pre-set using the environment variables positionned during the configuration. After checking that everything is coherent, press the button `init ros`.

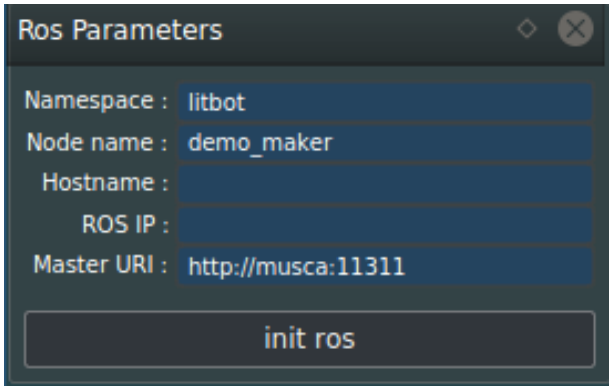

Figure 3: Widget used to configure the ROS middle-ware.

WARNING : Once the button `init ros` is pressed, you cannot modify the ROS configuration.

4 External Launcher

Some nodes should be started in a specific environment (GPU, robot...) and could not be directly launched in the host computer of the demo-maker.

The program named `demo_maker_external_launcher` allows to start some nodes from the demo-maker interface in remote computer. For that, you have to follow these steps:

1. Access the remote computer (called RC) using ssh connection.
2. Go to the litbot directory
3. Source the litbot package environment using :

```
$ source [litbot_dir]/ros_workspace/devel/setup.[bash, sh,
    ↪ zsh]
```

4. Position the environment variable `ROS_IP` if your RC IP address is not known by the DNS. This address should correspond to IP address of the current RC.

5. Position the environment variable `ROS_MASTER_URI` given in the output of the `roscore` command.

6. Start the `demo_maker_external_launcher` node.

```
$ rosrun litbot demo_maker_external_launcher <name>
```

   The name should be, by convention, the name of the RC (ex : musca, lito,...).

If everything goes well, you should see in the console :

```
$ rosrun litbot demo_maker_external_launcher alya
RECEIVE INVITE FROM MASTER
```

In the demo-maker interface, you could now choose this new computer in the launcher device list of launchers, under the name you passed in argument. When the interface requests to start a node, you should see in the console which nodes has been started by this external launcher and the command line used to start these nodes.

When the demo-maker interface is closed, all the nodes launched are terminated and the `demo_maker_external_launcher` program stays alive and waits for an invitation from the demo-maker GUI (another instance).

5 Configure nodes

Now, you need to configure the options for each launcher. This step should be done manually the first time and you can save the configuration into your computer for futur uses. If the configuration file already exists, go directly to the next section.

The parameters could be set in the widget `Node Properties` displayed in the figure 4. The nodes are organized by plugins.

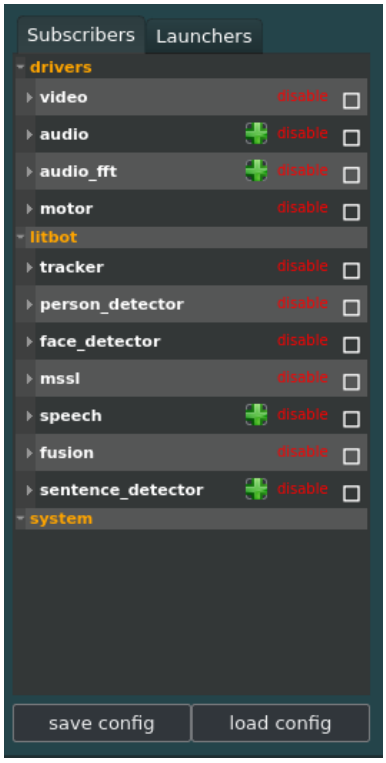

Figure 4: Widget used to modify launchers and subscribers parameters.

5.1 Prepare launchers and subscribers

In the demo-maker GUI, launchers configure the command line used to start a specific node, and subscribers subscribe to a topic in order to grab the output of the different nodes.

5.1.1 Launchers

Launchers contains information about the environment and allows to configure the command line used to start the node. You can choose on which computer you want to start the node under the widget `launch device`. The list contains the local launcher and all external launchers connected to the demo-maker. The `ros command, ros package, ros program and topic namespace` parameters are well configured by default and you do not have to modify them. These parameters are used in all launchers and show in figure 5.

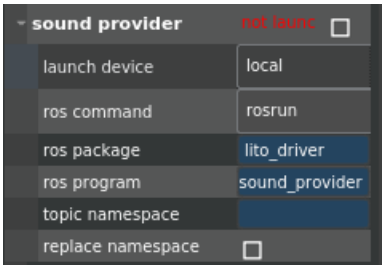

Figure 5: Parameters used in all launchers.

The end of this section will describe the options for each launchers. Please refer to the `litbot-documentation` for a node description. Parameters present in Figure 5 are presented above and will be no more explained in the next paragraphs. The nodes which do not have any additional parameters are not displayed here.

- **sound provider**

Parameter	**Type**	**Description**	**Value**
device	line edit	"Device name of the audio input. This is required only for LITO robot."	"plughw:0.0"

The device parameter could be found using the following command:

```
$ pacmd list-sources ...
```

and select the entry corresponding to the USB streamer.

- **video driver**

Parameter	Type	Description	Value
fps	spin box	"Rate of the image input (not working)"	15
device	line edit	"Device name of the camera. This parameter is required only for LITO robot."	/dev/video0
stereo dir	line edit	"Path to the stereo dir. This parameter is used only for NAO robot."	stereo_path

```
stereo_path = [litbot_dir]/global_workspace/
    ↪ external_independant_libraries/rmp/
    ↪ RobotSpecificPart/NAO/Properties/Stereo
```

The device parameter could be found using the following command:

```
$ v4...
```

and select the entry corresponding to the robot video device. This is used only for LITO robot.

- **audio resampler**

Parameter	Type	Description	Value
rate	spin box	"Target sampling rate of audio input"	16000

- **face detector**

Parameter	Type	Description	Value
face detector	combo box	"Choose the algorithm of the face detector. The CNN one use a neural network, and the PICO one is an algorithm not based on neural network."	"cnn"
with face features	check box	"Enable face features extraction. This parameter is enabled only if the face detector type is "cnn""	"true"
face finder	line edit	"Path to pico calibration. This parameter is enabled only if the face detector type is "pico""	pico_path

```
pico_path = <litbot_dir>/global_workspace/vision/
    ↪ face_detection/config/facefinder
```

If the face detector uses the cnn algorithm, it should be launched in a computer with GPU. Usually, we launch it on `alya`.

- **tracker**

Parameter	Type	Description	Value
enable siamese network	check box	"Use a siamese network for appearance model."	true
path to model	line edit	"Path to the appearance model."	appearance_path
tracking mode	combo box	"Choose the tracking mode. Visual performs only a visual tracking, VisualWithPartialAudio performs a visual tracking and associate the sound direction to the track, Full performs an audio visual tracking."	depends on the scenario

```
appearance path =
    Inside the repository :
        <litbot_dir>/global_workspace/vision/
            ↪ appearance_model
    Inside the install directory:
        <install_dir>/lib/python2.7/dist-packages/
            ↪ litbot/appearance_model
```

- **mssl**

Parameter	Type	Description	Value
audio calibration	line edit	"Use the audio calibration file. Click on the button D if you want to use elevation features. In this case, select the folder containing the grids. Click on the button F if you want to use only one elevation. In this case, select the calibration file. This value is specific to the robot"	mssl_grid

```
mssl_grid =
    for lito :
        <litbot_dir>/global_workspace/robot_config/
            ↪ mssl_calibration/LITO/elevation_grid
        <litbot_dir>/global_workspace/robot_config/
            ↪ mssl_calibration/LITO/single_grid/
            ↪ DPRTF_LITO_grid.txt
    for nao :
        <litbot_dir>/global_workspace/robot_config/
            ↪ mssl_calibration/NAO/elevation_grid
        <litbot_dir>/global_workspace/robot_config/
            ↪ mssl_calibration/NAO/elevation_grid/
            ↪ DPRTF_NAO_grid_XXcm.txt
```

- **behavior (fsm)**

Parameter	Type	Description	Value
disable motion	check box	"Disable motion of robot head in the fsm. This is used when the `name dialog system` node activated."	false

- **sentence detector**

Parameter	Type	Description	Value
node name	line edit	"Name used when connecting the node with ros master."	sentence_activity_detector
topic adjustment	line edit	"Modify the input topic on which this node will subscribe. The namespace is used for outputs. This option is used when you want to subscribe to another topic than the audio resampled one (output of the dereverberation....)"	""

- **speech recognition**

Parameter	Type	Description	Value
google credentials	line edit	"Credential used for google speech."	rely on the session/computer
node name	line edit	"Name used when connecting the node with ros master."	speech_recognition
topic adjustment	line edit	"Modify the input topic on which this node will subscribe. The namespace is used for outputs. This option is used when you want to subscribe to another topic than the audio resampled one (output of the dereverberation....)"	""

- **speaker_charac**

Parameter	Type	Description	Value
path to model	line edit	"Path to speaker charac model. This node should be activated only when tracker is in Full mode."	see node `main_speaker_charac` in litbot documentation

All of these configurations could be saved into a file. To generate default value for the different robots (when generating the configuration file), you can change the global parameter `robot` in the widget `Session Configuration`, and enter the configuration specific to this robot.

WARNING : Some unwanted effects are introduce when you save the configuration file and the selected robot is NAO. To avoid it, please select lito robot before saving the configuration.

5.1.2 Subscribers

Subscribers can be configured using the `Node Properties` widget, under the `Subscribers` tab. It contains the list of subscribers declared in the environment. Each subscriber connects to a specific topic. All subscribers have the following parameters:

- `topic name` : name of the topic the subscriber have to look for.
- `topic namespace` : additional namespace used for subscribtion. This is combined with the global namespace.
  The full namespace is `/global_namespace/topic_namespace/topic_name`.
- `replace namespace` : additional namespace used for subscribtion. This replaces the global namespace.
  The full namespace is `/topic_namespace/topic_name`.
- `display fps` : if this checkbox is checked, then the fps of this node will be displayed in the system visualizer.

By default, these parameters are correctly set, but you may need to change it for specific cases.

5.2 Session configuration

The panel `Session Configuration` contains two kind of things. The first one manages predefined selections of nodes, and the second lists some global parameters which are used by multiple nodes (see Figure 6).

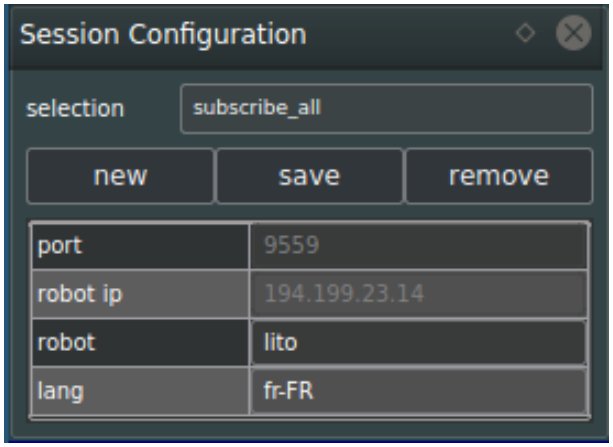

Figure 6: Widget used to configure the session.

5.2.1 Predefined Selection

When we show the demonstration, we follow different kind of scenario to handle different features of the algorithms. To facilitate the selection of the required node for a given scenario, you can predefine the required scenario and save it for future uses. The combobox contains a list of all predefined scenario, and the buttons `new, save and remove` manage the list of scenario.

To save the predefined scenaria permanently, you need to save the demo file (using `Ctrl-s` or the file menu).

5.2.2 Global parameters

The parameters stored in this widget are shared between all nodes. We can found:

- `port` : Port used for NAO communication (default : 9559)
- `robot ip` : IP address used for NAO communication
- `robot` : Select the robot you are using : lito/nao. The configuration will be automatically adapted for each robot.

- `lang` : language code used for speech recognition and name dialog system node.

6 Run the demonstration

In this section, we will explain how start/stop and manage the live demo.

6.1 Start/Stop the demonstration

The following methods can be used to start the demo in the demo-maker panel:

- You can click on the button `start simulation` present in the top tool bar (Figure 7, 1st button) or enter the keyboard shortcut `Ctrl+Space`.

- You can click on the button `start simulation [dual screen]` in the top tool bar (Figure 7, 2nd button), or use the key shortcut `Ctrl+Shift+Space`.

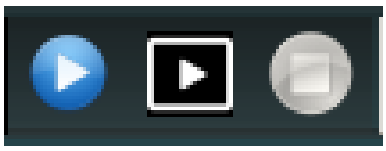

Figure 7: Buttons used to start/stop the demonstration.

When one of the previous method is used to start the demonstration, all the launchers selected by the user are launched in their associated computer. The subscribers start getting data from ROS topics and the surfaces start their update.

The dual screen is a floating window (not lock inside the demo-maker GUI) on which the scene of the current demo is displayed. The screens are controled independantly. The primary screen corresponds to the visualization inside the central widget of the interface, and the secondary screen

corresponds to the visualization inside the dual screen. This secondary screen is usually moved to the external monitor (TV, screen...) because it does not contains the configuration widget and the size allocated to display the scene is bigger than the one for central widget.

To stop the demonstration, you can :

- Click on the icon corresponding to this action (Figure 7, 3rd button)
- Use the key shortcut `Ctrl+Shift+e`

When the simulation is stopped, then all the started nodes are killed. If the demo is started in dual screen mode, it is not closed when the demonstration is stopped.

6.2 Manage viewers

In the demo files, you can create multiple representations of the algorithms corresponding to different scenario. Each scenario corresponds to a panel in the demo-maker scene and could be accessed using its name. To manage the scenario you want to display, you have to use the widget `Viewer Controller` (see Figure 8).

In this widget, the `primary screen` corresponds to the screen of the central widget, and the `secondary screen` corresponds to the viewer located in the dual screen. These two labels are used to select the screen you want to control. The label of the target viewer is colored in yellow. If you want to display the same region on both viewer, you can press the `Mirror` label. In this case, the area of the viewer selected before clicking on the label is replicated on the other viewer. When the `mirror` option is checked, both viewer labels are colored, and all actions are applied to both screen.

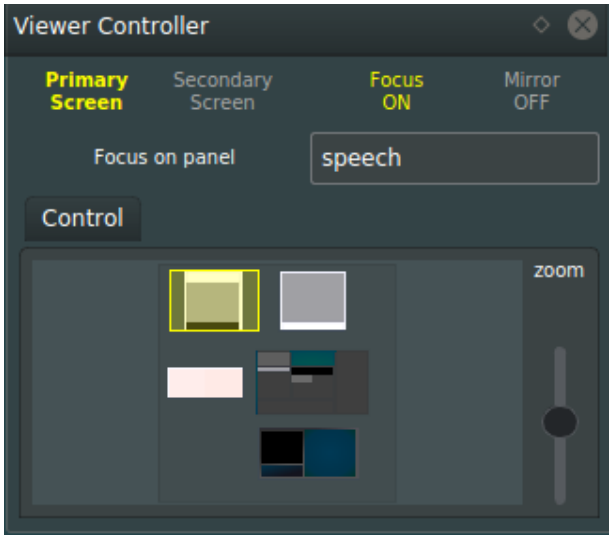

Figure 8: Widget used to manage the different viewers of the scene.

The different panels of the demo are registered in the combo-box `Focus on panel`. You can use this combo-box to focus on a specific panel in the selected viewer. The focus action consists of an optimization of the viewer area in order to have the highest size of the panel without modify its proportions. If the `Focus` label is enabled (ON), the transformation is done automatically. Otherwise, click on the label `focus`.

The group `Control` gives a representation of the full scene. A colored square item is added into the scene. These items represent the visible area of the activated viewer. The yellow one corresponds to the primary viewer and the second one to the secondary viewer. If you click and drag the square of the selected viewer, you can move inside the scene and the corresponding viewer will apply the transformations. The zoom item at the right of the scene permits to zoom on a specific region of the screen.

6.3 ROS parameter control

The algorithms of the project have some parameters relying on the environment in which you play the demo. You can dynamically adapt some of these parameters using the widget displayed in the figure 9.

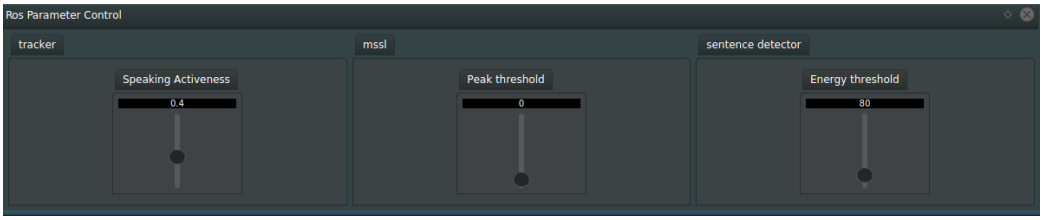

Figure 9: Dynamically adapt some parameters to the running environment, using the ROS parameters.

From the right to the left of the figure 9, you can find the following parameters:

- `Speaking Activeness` Threshold used by the `tracker` when the mode `VisualWithPartialAudio` or `Full` is activated. If the value is above this threshold, the track is considered as speaker. The threshold can be between [0 - 1] (default : 0.3)

- `Peak threshold` This threshold is used by the `mssl` when searching the peaks over the localization candidates. All peaks under this threshold are not take into account. It can be set between [0 - 1] (default : 0.1)

- `Energy threshold` This threshold is used by the `sentence activity detector`. It corresponds to the energy of a chunk from which the signal is considered as speaking. It can be set between [0 10000] (default : 80)

7 Visual element of litbot.demo panels

This section briefly explains the different panels defined in the litbot_new_debug.demo file.

7.1 Speech panel

The speech panel focus on the speech recognition results. It consists of three areas that you can see in the figure 10.

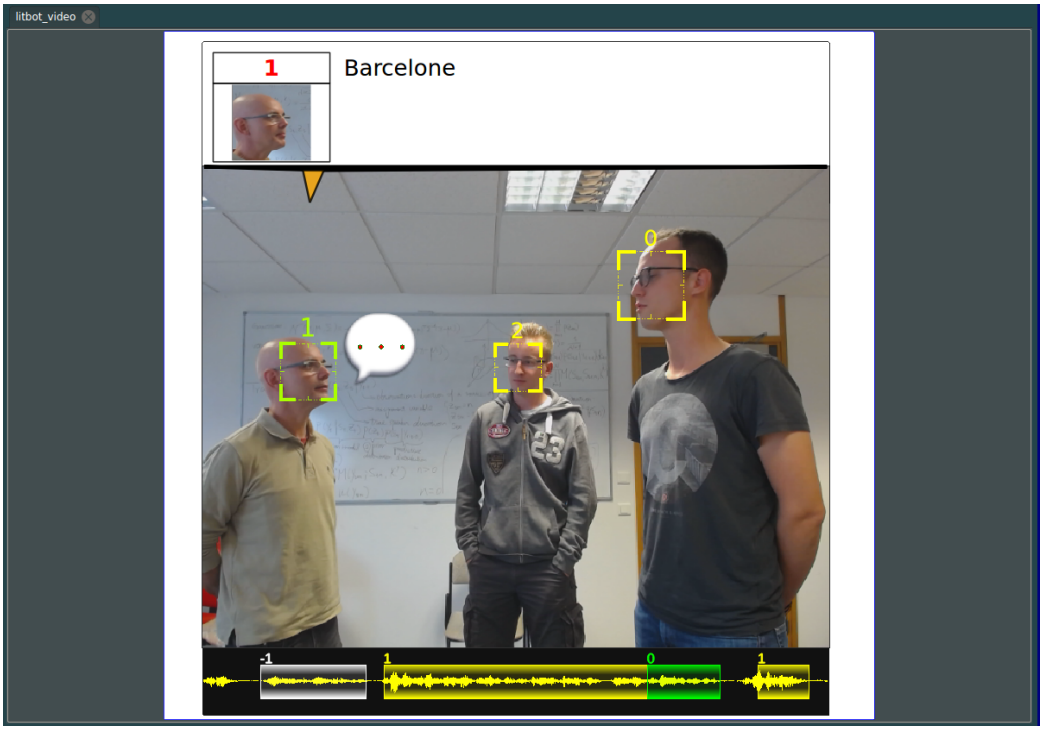

Figure 10: Demo maker speech scenario visualization.

- `Speech surface` : This area is designed to display the result of the speech recognition node. This contains the name and a picture of the speaker (when its face is visible), and the corresponding text. The cursor is here to point the speaker in the image.

- `Image surface` : In this surface, you can found the following items:
  - Image provided by the robot (left camera for NAO).
  - The red line corresponds to the projection of the sound source localization on the image.
  - Tracked faces are visualized with bounding rectangles. A bubble with dots is added when the user is speaking.

- `Audio surface` : In this surface, you can see one channel of the audio signal. The colored rectangle corresponds to sentence segmentation of the audio signal. One color is used for one ID. The content of each rectangle is sent to the speech recognition system (google speech).

If the speaker is not in the field of view of the robot, the id of the track is set to -1 in the speech surface. The colored rectangles in the audio surface are white and associated with the id -1.

7.2 Tracker panel

The tracker panel focuses on the result of the visual tracker. You can see two surfaces, the left one corresponds to the robot's image and the detected faces bounding boxes. The right one displays the robot's image and the tracked faces (the latent variables of the tracker and their corresponding id). The audio-visual fusion is not displayed in this panel. (See figure 11).

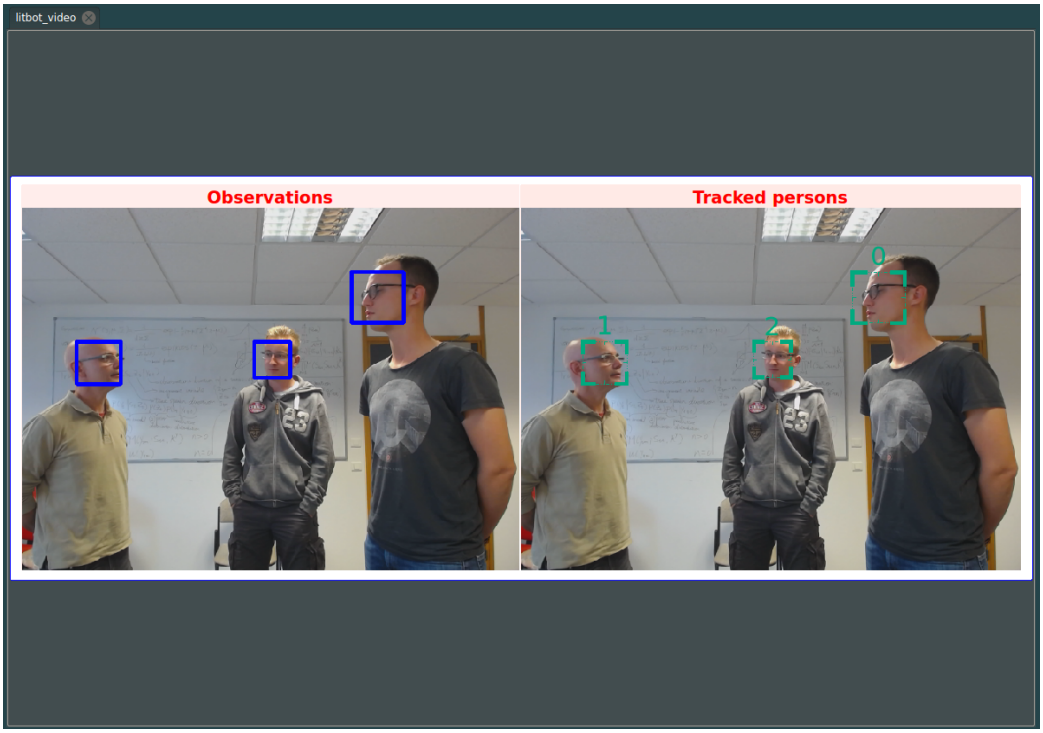

Figure 11: Demo maker tracker scenario visualization.

7.3 Mssl panel

This panel shows the mssl results (see Figure 12). It contains:

- The top surface contains the robot's image, the tracked faces, the projection of the sound source angle into the image (vertical red line), and the bubble associated to a track when it is considered as speaker.

- The second surface displays the sound source localization probabilities for each candidates along time. This is represented by a spectrogram.

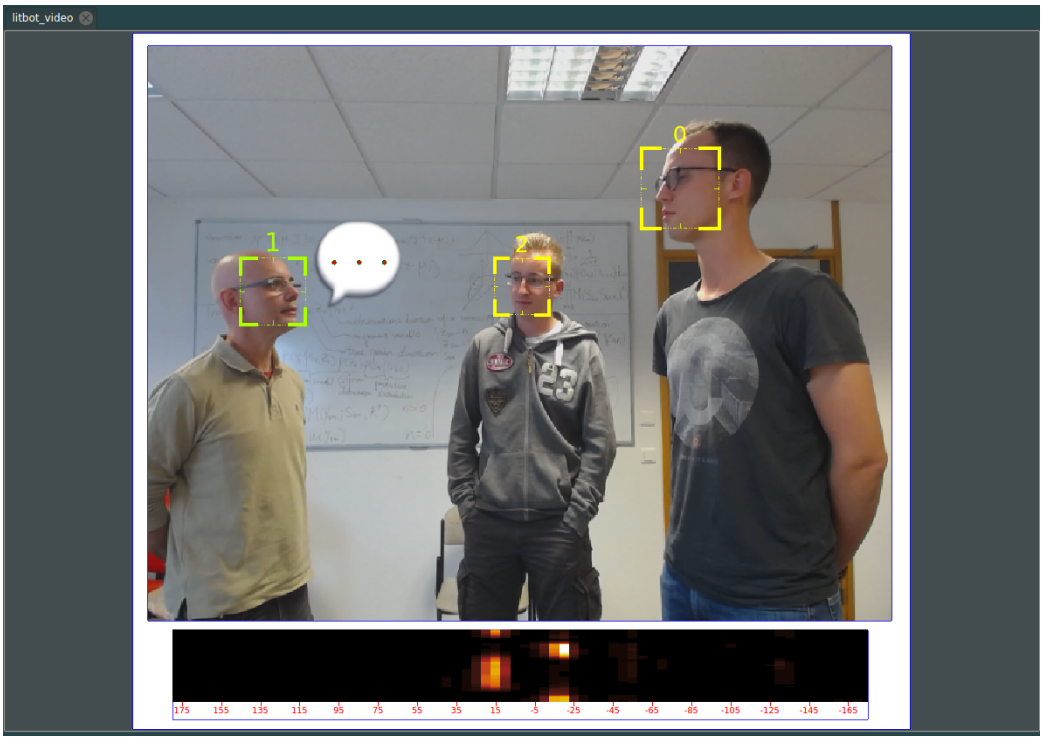

Figure 12: Demo-maker MSSL panel visualization.

7.4 Debug panel

The debug panel centralizes the visualization results of all algorithms and provides some information about the frequency rate, the CPU and memory usage of the host system. This panel is usually displayed in the primary screen but is not shown to user (see Figure 13)

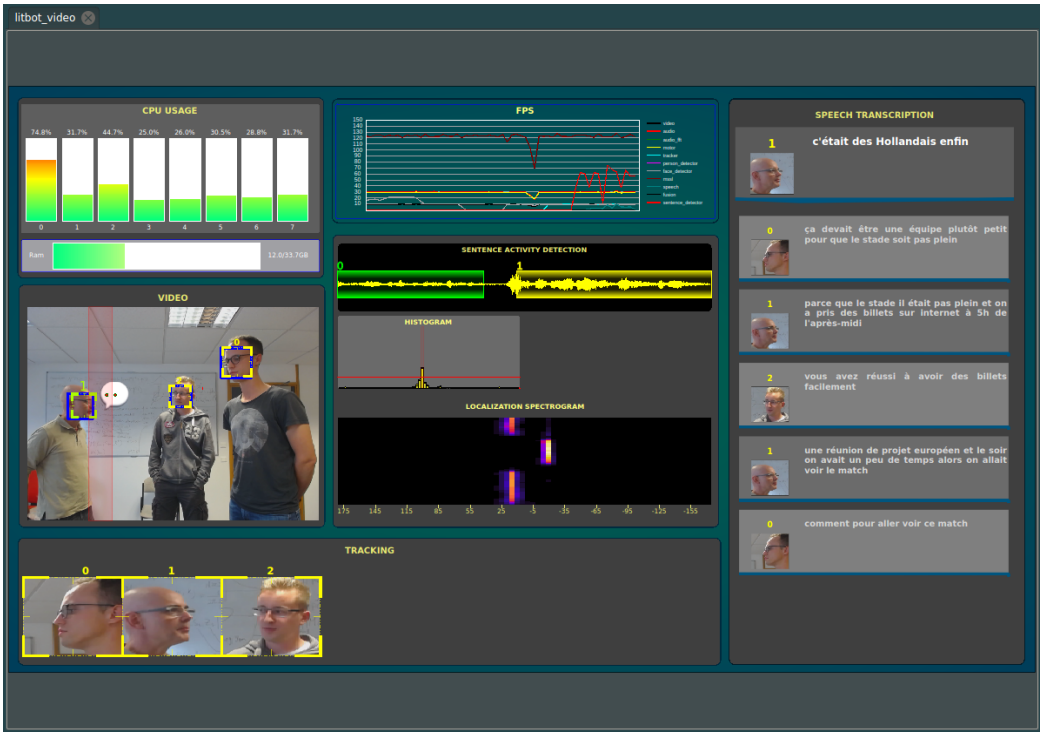

Figure 13: Demo-maker debug panel visualization.

You can find the following items :

- `CPU usage and RAM usage` : display information about the memory and CPU usage of the host machine using `htop` style representation.

- `Video` : in this surface, you can find the following items:
  - The robot's image
  - The detected faces bounding box (blue rectangles)
  - The tracked faces bounding box (colored rectangles associate to an id)
  - The projection of the sound localization in the image (vertical red line)

  - The audio visual fusion (bubble with dots closed to the speaking tracks).

- `Tracking` : in this surface, you can see an history of the track retained by the tracker node. A picture of each track is displayed.

- `FPS` : this surface shows a graphic representing the frequency rate of the nodes along time.

- `Audio` : this surface shows different informations about audio processing:

  - Sentence activity detector : audio time domain representation with colored rectangles corresponding to sentence segmentation of the audio signal.
  - Mssl histogram : show the probabilities of each candidate for the current timestep. The horizontal red line corresponds to the peak threshold from which the peak is retained by the algorithm. Then the bars of the histogram have several colors, that means that the MSSL is running using elevation informations. Each color corresponds to a specific elevation.
  - Mssl prior spectrogram : shows the probabilities of each candidates along time, using a spectrogram representation.

- Speech transcription : this surface displays the n-last results of the speech recognition system, in order to have an "history" of the conversation.

7.5 Top View panel

The top view panel is another global representation of the litbot system. It is mainly used when the behavior are activated and the robot moves its head (see Figure 14). You can find the following surfaces:

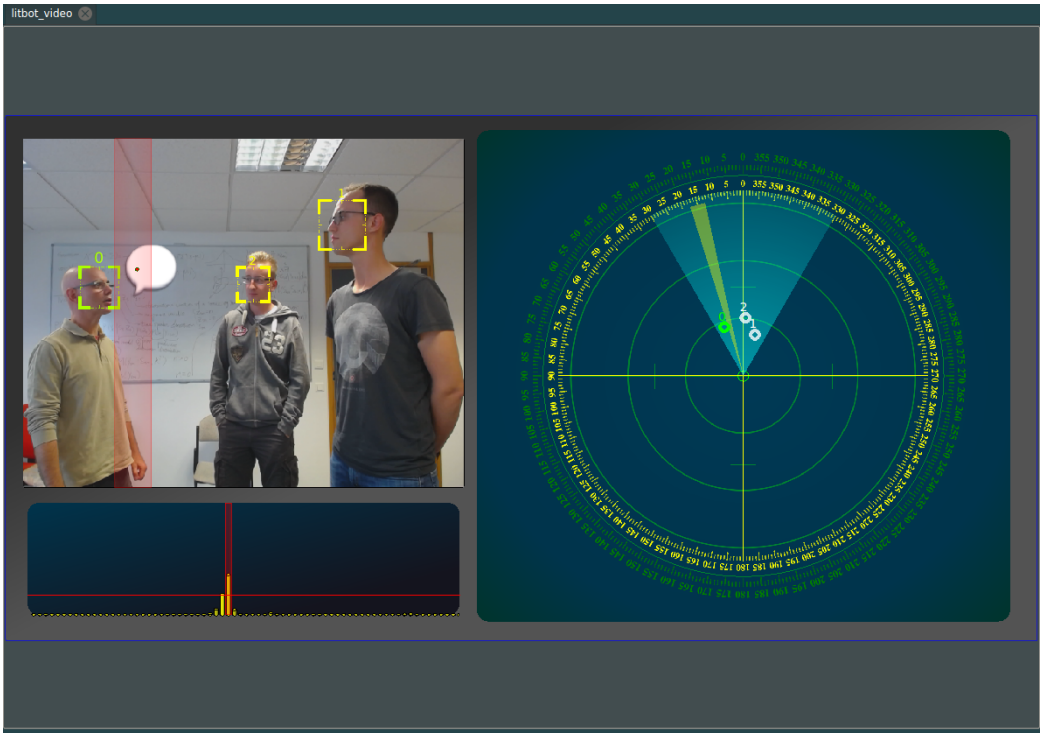

Figure 14: Demo-maker top view panel visualization.

- The first surface represents the robot's image, sound localization projection (red line), tracked faces and audio visual fusion.
- The second one shows the sound source localization histogram at the current timestep. The horizontal red line is the threshold from which the peaks are retained. The selected peaks are highlighted with vertical red line. The bars of the histogram could have different colors if the mssl is running with elevation features.
- The last surface represents a top view of the environment. You can found:
  - Concentric circles represent the distance from the robot (1m step) and the central point correspond to the position of the robot.

– The green circle with angles represents the angles in the fix coordinate.
– The yellow circle with angles represents the angles in the robot head coordinates. This circle is rotated according to the robot head position.
– The field of view of the robot is modelized by a rectangle area (highlighted with a lighter color). This triangle moves with the robot head.
– When a track is present, it is modelized by a small circle in the absolute position (fix over the head rotation).
– When a sound is located, its azimuthal angle is represented with a straight triangle in the target direction.
– The track circle changes its color when it is associated to a sound direction.