Create a ROS package that simulates a simple computational model of social human-robot interaction.

Detailed instructions follow

Scenario

Consider a social humanoid robot sitting at a table in front of a human child. At each interaction, a different colored toy is placed on the table, and the robot should express one emotion, which depends on both the perceived object properties and the perceived child behavior. This can be modeled with a Bayesian Network (see Fig. 1 below). The robot can perceive the size of the object (O), and classify it as either: small, big (they are all equally likely to happen). The robot can perceive human facial expressions (HE), and classify them as either: happy, sad, neutral (they are all equally likely to happen). The robot can perceive human head and eyes movements (actions, HA), and classify them as either: looking at the robot face, looking at the colored toy, looking away (they are all equally likely to happen). The robot face can express three possible emotions (RE): happy, sad, neutral. Note that, during an interaction, the robot might not have access to all the variables (object size, human expression, human action), due to e.g. absence of one of the stimuli or failure of a sensor, but still the robot has to decide what emotion expression is the most likely to be appropriate.

Part I --- The ROS Package --- [45 points]

• ***One ROS Package*** named ***cr_week8_test*** should be created, within the source folder of your catkin workspace (which should be something like $HOME/catkin_ws/src).

• The cr_week8_test package should depend on the following ROS packages: rospy; std_msgs; bayesian_belief_networks.

• Four additional folders must be created under the cr_week8_test folder: scripts; msg; srv; launch.

• ***Four ROS messages*** should be created.

• ***1) object_info***, which contains 2 discrete values (int), one for the id of the current interaction and one for the object size (1=small, 2=big): id, object_size.

Fig. 1: Proposed structure of the Bayesian Network.

• ***2) human_info***, which contains 3 discrete values (int): one for the id of the current interaction, one for the human action (1=looking at the robot face, 2=looking at the colored toy, 3=looking away) and one for the human expression (1=happy, 2=sad, 3=neutral): id, human_action, human_expression.

• ***3) perceived_info,*** which contains 4 discrete values (int): id, object_size, human_action, human_expression. If any of the perception variables is =0, it means that such information was not perceived during the "id-th" interaction.

• ***4) robot_info***, which contains 1 discrete value (int) for the id of the current

interaction and 3 real values (float) for the probability of each robot expression: id, p_happy, p_sad, p_neutral.

• ***One ROS Service*** called ***predict_robot_expression*** should be created, which takes as input one observation (object size, human action, human expression) and returns a probabilistic prediction about what emotion should the robot express, i.e. the probability of happy, probability of sad, probability of neutral.

• ***Four ROS nodes*** will need to be created:

• ***1) interaction_generator.py*** This node should generate data of a different interaction (object + human behavior) every 10 seconds. Each interaction has an *id* (1 for the first generated interaction, 2 for the second generated interaction, 3 for the third generated interaction, etc...), a random object size (1=small, 2=big), a random human expression (1=happy, 2=sad, 3=neutral) and a random human action (1=looking at the robot face, 2=looking at the colored toy, 3=looking away). The node publishes id and object size on a ROS Topic using the object_info message, and id, expression and action on a different ROS Topic using the human_info message. The Python random.uniform() function can be used to generate random values.

• ***2) perception_filter.py*** This Node subscribes to the ROS Topics created by Node 1, randomly filters the information, and publishes a perceived_info message on a ROS Topic. To filter the information, some of the variables (object_size O, human_action HA, human_expression HE) will be put to zero, which means they are not perceived during this interaction. To choose which variables will be put to zero (if any), a random integer between 1 and 8 must be generated, with the following meaning (1: O=0; 2: HA=0; 3: HE=0; 4: O=0,HA=0; 5: O=0,HE=0; 6: HA=0,HE=0; 7: O=0,HA=0,HE=0; 8: no modification - all variables available).

• ***3) robot_controller.py*** This Node should subscribe to the ROS Topic created by Node 2, and use the filtered information to infer the likelihood of each possible robot expression to be appropriate for the current situation (calling the "predict_robot_expression" service). If a variable is =0, it means it is not available (i.e. it cannot be used). The node should publish the id of the interaction and the probability of each robot expression on a ROS Topic using the robot_info message.

• ***4) robot_expression_prediction.py*** This Node implements the "predict_robot_expression" Service and makes it available to any external node requesting it. This should be realized with a Bayesian Network (using the bayesian_belief_networks package), with the structure suggested in Fig. 1 and the CPTs (Conditional Probability Tables) of Table 1. Note that the input observation should be allowed to be incomplete, e.g. object size not available or only human action available or nothing available.

• ***One ROS Launch file*** named ***human_robot_interaction.launch*** should be created that will start the four Nodes automatically.

Part II --- The README file

• A text file named **README.txt** that briefly explains the main steps needed to compile and use the package. The first line must include your name and student ID.

Part III --- The Video

• Take a 60 seconds screenshot video of your screen while you are running the different nodes and visualizing the probabilistic predictions about the robot emotion expression (generated by the robot_controller.py node).

• The screen should include:

- a window showing your readme.txt file, with your name well visible;
- the terminal from which you are running the Launch file;
- the ros_graph showing the nodes, topics and their connections;
- four separate terminals that show the ROS Topics published by Nodes 1, 2 and 3 (i.e. use the "rostopic echo" command) – 1 terminal for each Topic.

HE: Happy, Sad, Neutral; p(***HE***)=1/3.
HA: looking at Robot, Object, Away; p(***HA***)=1/3
O: Small, Big; p(***O***)=1/2.

HE	H	H	H	H	H	H	S	S	S	S	S	S	N	N	N	N	N	N
HA	R	R	O	O	A	A	R	R	O	O	A	A	R	R	O	O	A	A
O	S	B	S	B	S	B	S	B	S	B	S	B	S	B	S	B	S	B
RE =H	0.8	1	0.8	1	0.6	0.8	0	0	0	0.1	0	0.2	0.7	0.8	0.8	0.9	0.6	0.7
RE =S	0.2	0	0.2	0	0.2	0.2	0	0	0.1	0.1	0.2	0.2	0.3	0.2	0.2	0.1	0.2	0.2
RE =N	0	0	0	0	0.2	0	1	1	0.9	0.8	0.8	0.6	0	0	0	0	0.2	0.1

RE: Happy, Sad, Neutral; p(***RE|O,HE,HA***) = see table.

Table 1: Example values for the CPTs. In a real applications, these values might come from observations of human-human interactions (e.g. child-child or child-caregiver).