Telecom Paris **Class IA715-2021-2022**
G. VARNI – M. HULCELLE

Practical session 10/05/2022: Working on gesture expressivity

A serious game's company asks you to prototype a multimodal game for pre-school aged children for learning the associations between the expressivity of movement and sound qualities. The game is based on a multimodal dialog between the children and a humanoid robot. The game is structured so that only one child can interact with the robot in a room endowed with a Kinect sensor attached to a wall at 170cm from the floor and looking downward. A child enters the room and she has to stand in a game area of 4 x 4 meters in front of the Kinect sensor. The robot is placed just right under the sensor. The game is structured in the following way: there are 4 characters the child can play with: Airplane, Cook, Woodman and Kangaroo. The robot randomly chooses a sequence of them. The robot proposes the generated sequence to the child. To do that, it writes the name of the current character on the tablet it is endowed with. The child has to mimic a specific behavior of the chosen character using arms' gestures. For example, if the child chooses Airplane, she can mimic the flight of it by opening the arms and running straight into the game area. The robot observes the scene through Kinect: Kinect captures the gestures performed by the child and it sends the coordinates of the points of the arms' of skeleton of the child to the robot. It draws the trajectories of some points of the arms on its tablet and analyses these trajectories through the Laban Movement Analysis (LMA). This is done to extract the expressivity of the movement performed. If the child successfully mimics the behavior of the current character, the robot proposes her the next one. Otherwise, it waits until the child becomes successful. At the end of the sequence, the robot plays a sequence of sounds having qualities reflecting those ones detected in the movement performed. In particular, the designer of this game implemented the following associations between movements and characters.

Character	**Movement**
Airplane	Direct and Quick
Woodman	Direct and Sustained
Cook	Flexible and Sustained
Kangaroo	Flexible and Quick

In this practical session, you have to develop a simplified proof-of-concept of this dialog system between the child and the robot. You will simulate the arms' trajectories of the child by using trajectories drawn on the mousepad using the mouse as an arm. The laptop will simulate both the Kinect sensor and the robot: that is, it will acquire the trajectories, it will perform the LMA and the mapping onto sound. Finally, it will generate the answer.

Open the **Python 3** TP_*Laban.ipynb* notebook on **Anaconda.**

In more details:
- the user runs the TP_*Laban.ipynb* notebook: a white window with two buttons appears;
- to start the game, the user presses the "Start_game" button. The selected sequence of characters appears written on the main console;
- the user can now start to draw trajectories. As soon as she clicks on the window to draw the name of the character to be mimicked appears in the top right corner of the window;
- once the user has done, the sequence of the detected characters is printed on the screen (this simulates the play back of the audio sequence).

At work!
Look at the main program you will find at the end of the file and at the *create_widgets* function. This creates a canvas, which is contained in the window displayed on the tablet. This includes a "Start game" button to start the game and a "Delete" button to clean the canvas. When the user presses the "Start game" button, the *start_analysis* function is called that generates the current sequence of characters by means of the *generate_characters_sequence*.

1. Implement function *generate_characters_sequence*.

Each time a new point belonging to the 2D trajectory of the mouse is captured, the *paint* function is called, which draws the trajectory on the canvas. Points are accumulated in a buffer to perform analysis.

2. How is such buffer managed? Which events make analysis start?

The *paint* function also manages the transitions between a character and the next one, by printing the name of the new character on the canvas. Finally, it calls the *synthesis* function as soon as the sequence ends, thus enabling play back of the audio content which closes the interaction. Here the playback is simulated by printing the sequence of the detected characters on the screen.

3. Implement the *synthesis* function, printing a suitable string.

Function *analyse_continuous_data* invokes function *laban_analysis*. This takes the buffer containing the input trajectory and computes Laban's Space and Time features.
For the Space factor of Laban's Effort, you can use the directness index and the goodness of fit (R^2) of a linear regression computed on the trajectory.
The directness index is computed as:

$$DI = \frac{d(p_1, p_N)}{\sum_{k=1}^{N-1} d(p_k, p_{k+1})}$$

where p_i are the points of the gesture's trajectory, and in particular, p_1 and p_N are the first and the last point of the trajectory, *d* is the Euclidean distance.
Concerning regression, assume the *x* coordinate as the regressor, and *y* as the dependent variable. To compute a regression look at the *linear_model* module in the *sklearn* library. To compute R^2 look at the *sklearn.metrics* module.

4. In order to implement Laban's Space factor, fill the body of the following functions: *directness_index, goodness_of_linear_fit*.

For the Time factor of Laban's Effort, you can compute impulsiveness and the normalized average of kinetic energy.
Let $s_1, s_2, \ldots s_n$ be the values of speed computed at sample 1, 2, … *n* (being *n* the number of samples the trajectory consists of). Then, normalized average of kinetic energy is computed as:

$$ANKE = \frac{1}{n * \max_i s_i^2} \sum_{i=1}^{n} s_i^2$$

This is already available in the notebook.
Impulsiveness is computed as:

$$Imp = \frac{\max_i s_i}{n}$$

Since *Imp* belongs to [0, +∞), it needs to be normalized so that values in between 0 and 1 are mapped onto [0, 0.5] and values in between 1 and +∞ are mapped onto (0.5, 1).

5. In order to implement Laban's Time factor, fill the body of the *impulsiveness* function.

The output of *laban_analysis* is given to the *which_character* function, which computes the index of the character corresponding to the computed Laban's features. The index of the character is appended to a buffer, which contains the last identified characters. After 5 buffers are processed, the *who_is_mimicked* is invoked, which performs *majority voting* and returns the most voted character.

6. Implement the *who_is_mimicked* function.

If the function returns is the correct character, the game moves to the next character in the sequence by setting variable *character_changed* to 1, so that the name of the next character will be displayed as soon as the *paint* function is called again.

As it is implemented now, the robot waits until the child becomes successful. This strategy, however, can be very stressful and frustrating if a child cannot mimic one character properly.

7. Modify the *analyse_continuous_data* function so that if after 3 attempts the child cannot still mimic a character successfully, the game moves anyway to the next character.

Consider the *Interaction manager* module of the W3C framework for multimodal interaction.

8. Which turn tacking strategy does this system implement? Briefly discuss pros and cons of this choice and identify possible directions for improving interaction with the system.