Enriching Robot's Actions with Affective Movements

Julian M. Angel-Fernandez[1] and Andrea Bonarini[2]

Abstract— **Emotions are considered by many researchers as beneficial elements in social robotics, since they can enrich human-robot interaction. Although there have been works that have studied emotion expression in robots, mechanisms to express emotion are usually highly integrated with the rest of the system. This limits the possibility to use these approaches in other applications. This paper presents a system that has been initially created to facilitate the study of emotion projection, but it has been designed to enable its adaptation in other fields. The emotional enrichment system has been envisioned to be used with any action decision system. A description of the system components and their characteristics are provided. The system has been adapted to two different platforms with different degrees of freedom: Keepon and Triskarino.**

I. Introduction

The development of fast, cheap, and reliable electronics has enabled the creation of new devices and versatile robotic platforms. These new platforms' capabilities have expanded the frontiers of the robotic applications to new environments, where robots are expected to interact with humans, such as health care and house cleaning, among others. However, bringing robots in these environments raises the challenge to increase robots' acceptance. Although this could be seen as an easy task that just would need improvements in robots' appearance and capabilities, it is possible that people would expect to treat robots as humans, like they do with computers. [1], which makes necessary the creation of robots able to fulfil these expectations.

Some researchers have suggested that embedding emotion expression capabilities in robots could improve their acceptance in social environments [2]. To this purpose, some researchers (e.g., [3], [4]) have added specific emotional poses and expressions to their robots. Others have studied how to convey emotions with specific platforms [5], [6]. Nevertheless, these works have created modules to show emotions that are strongly integrated in their systems, which eliminate the possibility to re-use or adapt them for other projects.

In theory, the projection of emotion with humanoid embodiments could be simplified by mimicking human movements. However, this cannot be fully implemented, due physical limitations in robots. Since an exact matching of human movements to convey emotions could not be used in robots [7], [8], some researchers are studying diverse features and values to express emotions with different platforms. As a consequence, these results could not be widely used due to the difference among the platforms.

This paper presents an Emotional Enrichment System (EES) that modifies actions' parameters and adds actions to create the illusion of emotion expression in a robot. Although the EES was originally conceived to be used to facilitate the study of emotion projection, its design was devised to make it extendable to other platforms and adaptable to new tasks. To achieve this goal, the system relies on an Emotional Execution Tree (EXT), which is based on simple actions, sequential and parallel nodes. Additionally, the concept of compound actions is used to group a bunch of nodes, thus reducing the tree dimension and allowing the reuse of recurrent actions generated by specific combinations of simple actions and other nodes. This EXT has been formalized to provide a guideline to further implementations and extensions.

The rest of the paper is organized as follows. Section II provides a brief overview of relevant work. Section III introduces all the concepts that are used in the system. Section IV gives the basic formalization of our system and principal components terms used on it. Section V presents the EBFN created to specify the files used by the system. Section VI describes the implementation of the system and shows two demonstrations done with the system using platforms with different capabilities.

II. Related Work

The use of emotion enrichment to improve human robot interaction is not new. There have been several researchers that have enhanced their social robots with emotions or studied how to convey emotions in robotic platforms [5], [6]. One of the first, well-known expressive robots is Kismet [3], a robotic face able to interact with people and to show emotions. This platform uses a specific set of movements based on the Ekman's studies on human emotion expression [9]. Other approaches have tried to use anthropomorphic platforms [4] to convey emotions to study the response of people towards the robot. However, the emotions portrayed were hand-coded, hard-wired to the respective platforms, and their parametrization is not available.

On the other hand, studies focused on entertainment robotics have tried to introduce emotional actions to improve the audience's experience. Breazeal and collaborators [10] used one robot on the stage. Their anemone-like robot had few behaviours, which included getting scared when a person comes too close; the robot was able to show some basic emotions (i.e., fear and interest). Knight [11] used the platform NAO to produce a sort of stand-up comedy. The robot performs basic actions to add some expressiveness to the

[1]Julian M. Angel-Fernandez is Researcher at Automation and Control Institute, Vienna University of Technology, Vienna, Austria julian.angel.fernandez@tuwien.ac.at

[2]Andrea Bonarini is Full Professor at the Department of Electronics, Information, and Bioengineering, Politecnico di Milano, Milan, Italy, andrea.bonarini@polimi.it

joke, but it is not intended to project any emotion. Trying to add some theatrical realism, Breazeal and collaborators [12] designed and implemented a system to control a lamp. The main characteristic of this lamp is that it could be controlled by just one person, by selecting pre-coded emotions.

Other works in performance robotics have developed systems that do not convey any kind of emotion as *Roboscopie* [13], Fan and collaborators [14], and adaptation of Shakespeare's Midsummer Night's Dream [15] use robots in performances with real actors, but without any emotional expression. Although these works have expressed emotions in their studies, they did not focus on creating a framework that could be used by other researchers in HRI. The use of affective architectures for virtual agents (e.g., SAIBA/BML framework [16]) could be thought as a feasible solution. However, these frameworks describe high level actions without giving relevant details to its implementation in physical platforms. This gap cannot be filled by making a direct mapping between the abstract actions and the physical platforms. This is due to the fact that virtual agents do not have physical constraints, while robots are constrained by their physical capabilities [7], [8].

III. Basic Concepts

The system is based on six main concepts that structure and enable the enrichment of actions with emotions. (i) *Action message* establishes the structure of the message to describe any kind of action (i.e., simple and compound). This message also specifies how the actions are executed (i.e., in parallel or in sequence) and which action is predominant (i.e., primary or secondary). (ii) *Emotional parameters* describe how the emotional enrichment should be done to convey a specific emotion in a specific simple action. This description could also include addition of other simple actions and vary over time. (iii) *Character description* enables the possibility to establish how to modify emotional expressions to generate diverse "rhythms". (iv) *Emotional Execution Tree* (EXT) is a computational representation of desired actions that should be executed. This tree is created from the action message description and then modified using the emotional parameters and character description. The remaining two concepts rquire amore detailed description, reported below.

A. Emotion Parameters

The emotion parameters specify how actions parameters should be modified in order to express an emotion. These parameters could include a sequence of descriptors. For example, these descriptors could be used to generate changes of velocity or position over time to express an emotion. However, there is going to be a diverse set of parameters and descriptors depending on the nature of the action. For example, consider the actions *move body* and *speak*. The human action of speaking is related to changes in the vocal cords to produce different sounds, while *move body* is directly related to the body movement. The speaking action would have the following parameters: text, pitch, and tone. On the other hand, the action *move body* would have as parameters the desired destination, velocity, and trajectory constraints. However, in order to express emotion in speak action, it is required to modify pitch variables (i.e., fundamental frequency level, contour and jitter), intensity and speech rate [17]. On the other hand, changing linear velocity and adding oscillation could be used to convey emotion in the move body action [18]. Since each action requires modification of diverse parameters, a variety of emotional parameters would be necessary to describe the modification of all types of actions. So, it was decided to focus on the parameters that could be used to express emotions with the robots we have considered: Triskarino and Keepon. It was determined that these parameters could just use movement actions, which are described by position, velocity and acceleration.

B. Simple and Compound Actions

In order to generate a system that could be used in diverse platforms, an abstraction level in which all of them could fall is required. Simple and compound actions are used to achieve this goal. Simple actions are actions that are considered as primitives: they are used as building blocks. As a consequence, these actions are described in the system and are the ones whose parameters are changed to embed the emotions. Their description specifies mandatory and optional parameters that are required to execute an action. These parameters are modified accordingly to the information provided by the emotional parameters. Compound actions are actions that are created from simple actions and other compound actions. For example, two actions that are executed in parallel are considered as compound actions. These actions are not always implemented in the system. They can be described in the action messages without being implemented. If a compound action is used often, it could be implemented in the system. The implementation describes action's parameters, other compound actions, simple actions, and mapping between actions and parameters.

A set of simple actions to be implemented were selected to test the system, considering the capabilities of Keepon and Triskarino. The eight actions selected were: *move body*, *oscillate body*, *move shoulder*, *oscillate shoulder*, *move torso*, *oscillate torso*, and *do nothing*. Description for each action, its mandatory parameters and optional parameters are presented in Table I.

To exemplify the use of these actions, let us consider a sequence of movements illustrated in Figure 1. This sequence of movements corresponds to an adaptation of the first part of the balcony scene of Shakespeare's Romeo and Juliet play [19]. The fist part of this scene is characterized by strong changes in Romeo's emotional state, thus it was selected as a significant test for the system. However, just body movements would not exploit the other movements of the platform neither test the whole system. Thus, additional movements are added to illustrate and test the system. A partial view of the whole sequence of actions is illustrated in Figure 2. The first node corresponds to the first action in Figure 1. The second node corresponds to a compound

TABLE I: Description of the seven simple actions implemented, and their respective parameters, where P is 2D position, V is 2D velocity vector and angular velocity, and T is time

Action Name	Description	Parameter(s)
Do nothing	It waits for a time t before it is terminated. It could be seen as a delay.	T
Move body	It moves the platform from its current position a to a desired position b.	P, and V
Oscillate body	It generates an oscillation in the whole platform by an angle θ.	θ and V
Move shoulder	It moves the shoulders to a desired angle θ. It is considered as angular movement.	θ and V
Oscillate shoulder	It oscillates the shoulders by a given angle θ	θ and V
Move torso	It moves the torso to a desired angle in yaw, $pitch$ and $roll$	yaw, $pithc$, $roll$ and V
Oscillate torso	It oscillates the shoulders by a given angle θ	θ and V

action created by the parallel execution of the simple action *move body* and other compound actions.

IV. Emotional Tree

Emotional Execution Tree is a connected acyclic graph $G(V, E)$ with $|V|$ vertexes and $|E|$ edges. The root and non-leaf nodes could be of either *parallel* or *sequential* type, which represent the type of actions that would be executed by the system. Sequential nodes execute one branch after the other. Once all the branches have been executed, a sequential node notifies its predecessor to continue with the execution of other branches. The parallel node executes all the branches at the same time. This type of node offers two levels of synchronization: action and emotion. The action level synchronization enables the possibility to stop all the branches once the main branch has completed its execution. This main branch is established in the action message. On the other hand, the emotion level synchronization makes all branches to shift to the next descriptor of the emotion. In both cases, any update or finish message will be ignored if the branch is not principal. These two levels of synchronization create four sub-types of parallel nodes: (i) action and emotion synchronous, (ii) action synchronous and emotion asynchronous, (iii) action asynchronous and emotion synchronous, or (iv) action and emotion asynchronous.

Finally, leaf nodes are only simple action nodes that have been implemented in the system. Any node can belong to one of two levels: principal or secondary. If a node is principal, it will notify its predecessor about the messages that it has received, while the secondary node cannot propagate any message to its predecessor. This enables the possibility to have actions that have priority over others. To exemplify this, let us consider the sequence of actions depicted in Figure 2. The Emotional Execution Tree for this sequence would be like the one presented in Figure 3.

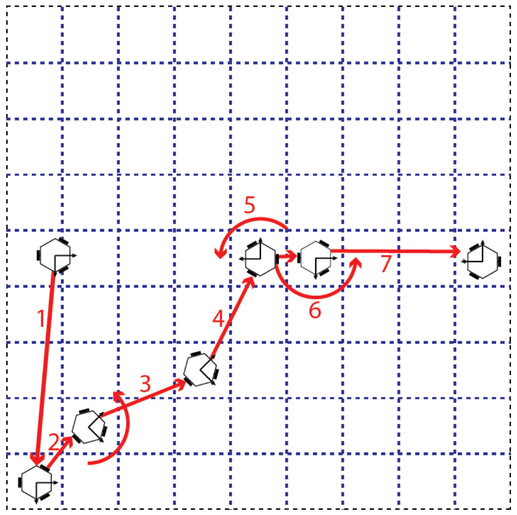


Fig. 1: Sketch of Romeo's movements for the balcony scene in Romeo and Juliet play. The arrows show the direction of the movements, and the numbers their position in the sequence.

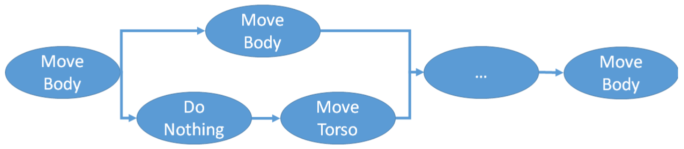


Fig. 2: Sequence of simple actions for the movements depicted in Figure 1.

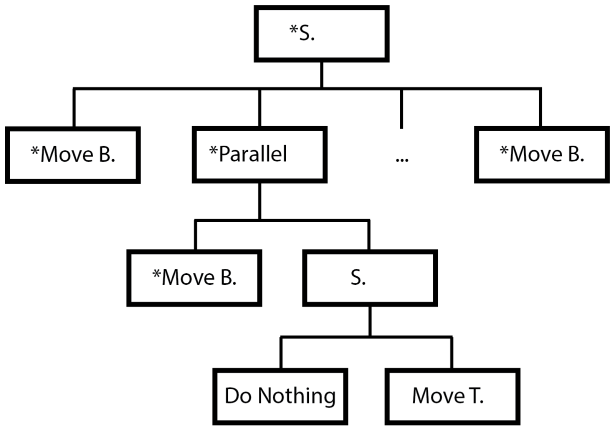


Fig. 3: Emotional Execution Tree for the sequence of actions depicted in Figure 2. The $*$ symbol denotes principal nodes. S represents sequential.

A. *Formalization of Simple and Compound Actions*

Formalizing simple and compound actions enables the generation of a system to design them and verify if they are described correctly. Simple actions (SA) could be seen as functions that map a set of parameters $p_i^{pl} \in P$ to specific platform (pl) movements ($m_j^{pl} \in M^{pl}$) in the set of movements for that platform ($m_j^{pl} = sa^{pl}(\{p_i^{pl}\})$). Each sa^{pl} could have different implementations due to different reasons such as: platform's configuration, action's goal, how the mapping from inputs to outputs are done, among others. In our case, we consider that two actions are different if $(\forall sa_i, sa_j \in SA, \forall q, t \in PL | Param(sa_i^q) \neq Param(sa_j^t) \Rightarrow sa_i^q \neq sa_j^t)$, where PL is the set of all the possible platforms, $Param(\cdot)$ returns the set of parameters that are input of the sa (e.g., vel, $pitch$, $text$). On the other hand, we will consider two actions as equivalent if $(\forall sa_i, sa_j \in SA, \forall q, t \in PL | Param(sa_i^q) = Param(sa_j^t) \wedge Obj(sa_i^q) = Obj(sa_j^t) \Rightarrow sa_i^q \equiv sa_j^t)$, where $Obj(\cdot)$ returns the set of objectives that are intended to be achieved by a given action. This SA equivalence allows us to avoid the specification of all the platforms for all the actions that share the same objective and hide leaf nodes that do not give relevant information to understand the emotional tree.

With this SA definition, it is now possible to define emotion enrichment as $(\forall sa \in SA, \forall e \in E, \exists\ en \mid en = Enrichment(sa, e, Param(sa), Intensity(e)))$, where $Intensity(\cdot)$ returns the intensity of a given emotion, E is the set of emotions, and en can be the same sa, but with modified parameters in order to express the desired emotion e, or even a set of sa with modified parameters. This definition goes along with the idea that new sa could be added to convey a specific emotion. In addition, the $Enrichment(\cdot)$ function has the following properties:

- $(\forall sa_i, sa_j \in SA, \forall e \in E \mid sa_i \equiv sa_j \Rightarrow Enrichment(sa_i, e, ...) \equiv Enrichment(sa_j, e, ...))$
- $(\forall sa_i, sa_j \in SA, \forall e \in E \mid (sa_i \not\equiv sa_j) \Rightarrow Enrichment(sa_i, e, ...) \not\equiv Enrichment(sa_j, e, ...))$

The concept of compound action (CA) is defined as $g = ca(\{p_i\})$, where $(p_i \in Param(sa))$, $(sa \subseteq SA)$, $(g \in EXT)$.

V. Describing Components

Description files allow the parametrization and extention of the system to fit other applications. A language was created to define the content of these files. This allows other researchers to create their own files and to extend the language to fit their needs. The Extended Backus-Naur Form (EBNF) that describes this language is reported below.

A. *Emotion Description*

The language created to describe emotion descriptors is the following:

⟨*emotion description*⟩→ ‘‘{” ‘‘**emotion:**” ⟨*string*⟩‘‘,” ‘‘**observation:**” ⟨*string*⟩‘‘,” ⟨*action*⟩ (‘‘,”⟨*action*⟩)* ‘‘}”

⟨*action*⟩→⟨*string*⟩‘‘**:** {” ⟨*description*⟩‘‘,” ⟨*actions affected*⟩ ‘‘}”

⟨*description*⟩→ ‘‘**description:** {” ⟨*action name*⟩‘‘,” ⟨*emotion*⟩‘‘,” ⟨*parameter's type*⟩ ‘‘}”

⟨*action name*⟩→ ‘‘**emotionProfileAction:**” ⟨*string*⟩

⟨*emotion*⟩→ ‘‘**emotionProfileEmotion:**” ⟨*string*⟩

⟨*parameter's type*⟩→ ‘‘**movement_parameter**”

⟨*actions affected*⟩→ ‘‘**actions:** {” ⟨*action parameter*⟩ (‘‘,” ⟨*action parameter*⟩)* ‘‘}”

⟨*action parameter*⟩→ ⟨*string*⟩‘‘:{” ⟨*reference*⟩‘‘,” ⟨*repetition*⟩‘‘,” ⟨*parameters*⟩ ‘‘}”

⟨*reference*⟩→ ‘‘**reference:**” ⟨*number*⟩

⟨*repetition*⟩→ ‘‘**repetition:**” ‘‘**yes**” | ‘‘**no**”

⟨*parameters*⟩→ ‘‘**parameters:** [” ⟨*parameter description*⟩ (‘‘,” ⟨*parameter description*⟩)* ‘‘]”

⟨*parameter description*⟩→ ⟨*movement parameter description*⟩

⟨*movement parameter description*⟩→ ‘‘{**time:**” ⟨*number*⟩ ‘‘, **space:**” ⟨*number*⟩ ‘‘}”

The attribute repetition is used to specify if the descriptors of an emotion parameters must be executed in an infinite loop. As it was already mentioned, the emotional parameters implemented for this version correspond to the move actions.

B. *Action Message*

The language created to describe action Messages is the following: ⟨*action*⟩→ ⟨*simple action*⟩ ⟨*compound action*⟩ ⟨*context*⟩

⟨*simple action*⟩→ ‘‘{” ⟨*action header*⟩‘‘,” ‘‘**parameters:** [” ⟨*simple action parameters*⟩ ‘‘] }”

⟨*compound action*⟩→ ‘‘{” ⟨*action header*⟩‘‘,” ‘‘**parameters:**[” ⟨*compound action parameters*⟩ ‘‘]}”

⟨*action header*⟩→ ‘‘**type:**” ⟨*action type*⟩‘‘,” ‘‘**name:** ” ⟨*string*⟩‘‘,” ⟨*is primary*⟩

⟨*simple action parameters*⟩→ ‘‘{” ⟨*parameter header*⟩‘‘,” ⟨*parameter description*⟩ ‘‘}”

⟨*compound action parameters*⟩→ ⟨*simple action parameters*⟩ (‘‘,” ⟨*simple action parameters*⟩)*

⟨*context*⟩→ ‘‘{” ⟨*action type*⟩‘‘,” ⟨*emotion sync*⟩‘‘,” ⟨*action sync*⟩‘‘,” ⟨*is primary*⟩‘‘,” ‘ ’**information:**” ⟨*string*⟩‘‘,” ‘ ’**actions:** ” ⟨*action*⟩ ‘‘}”

⟨*parameter header*⟩→ ‘‘**type:**” ⟨*parameter type*⟩‘‘,” ‘ ’**name:**” ⟨*string*⟩

⟨*parameter description*⟩→ ⟨*parameter amplitude*⟩ | ⟨*parameter circle*⟩ | ⟨*parameter landmark*⟩ | ⟨*parameter point*⟩ | ⟨*parameter speech*⟩ | ⟨*parameter square*⟩ | ⟨*parameter time*⟩

⟨*parameter type*⟩→ ‘‘**mandatory_parameter**” | ‘‘**optional_parameter**”

⟨*is primary*⟩→ ‘‘**yes**” | ‘‘**no**”

⟨*emotion sync*⟩→ ‘‘**yes**” | ‘‘**no**”

⟨*action type*⟩→ ‘‘**parallel_node**” | ‘‘**sequential_node**” | ‘‘**simple_action**” | ‘‘**composite_action**”

VI. Implementation

The system was implemented in *C++* and interfaced with ROS. The design (Fig. 4) was created following the description presented in the previous sections. The emotion enrichment core is divided in three different modules. Each module is responsible for one of the following phases:

1) *Generation of emotional execution tree:* this phase starts every time that a new action message is received.

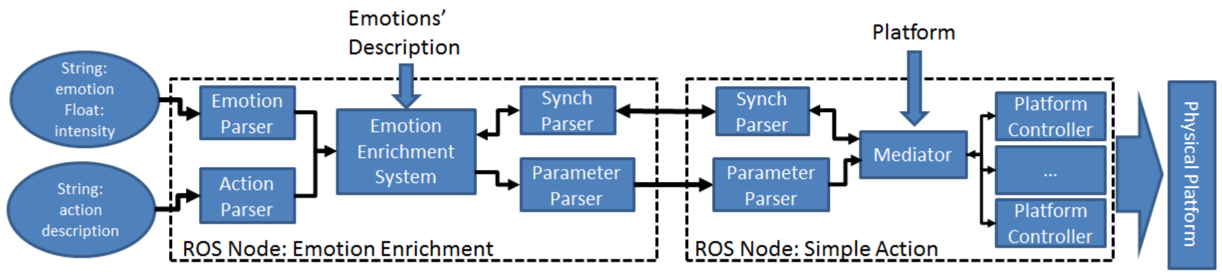


Fig. 4: General system design. Each simple action corresponds to one ROS node, and there is just one node for the emotion enrichment system. The ovals represent the ROS topic parameters, rectangles represent *black boxes*, and texts outside containers represent input files that contain the system parametrization.

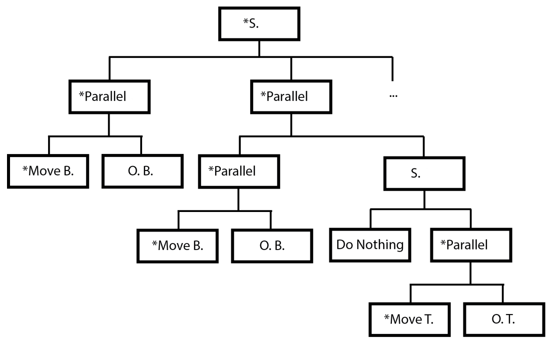


Fig. 5: EXT enriched with emotion for the EXT presented in Figure 3.

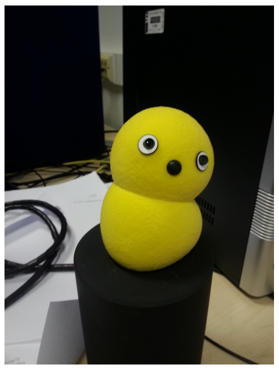


Fig. 6: Keepon platform.

The process begins by parsing the format, verifying that the actions described on it exist in the system, and that the parameters correspond to the ones expected by each action included in the message. When the verification is done, and all the actions exist and the parameters correspond, an EXT such as the one presented in Figure 3 is created.

2) *Emotion addition:* uses the EXT created in the previous phase. In this phase new simple actions (sa) can be added to the EXT, and the sa's parameters are modified following the emotion description, which is loaded from files. This process is broken down in two steps. First, all the actions that are required to convey the desired emotion, and that are not yet present are added. Second, the emotional parameters are modulated basing on the emotion's intensity and character traits. The final EXT is presented in Figure 5.
3) *Execution:* this is the last phase and it is done after the EXT is "coloured" with emotional features (actions additions and emotional parameters). The decision to have two different communication channels, one for action parameters and another for the action emotional parameters, was taken to enable the possibility to update the emotional parameters without interfering with the current execution.

All the text message broadcast among the nodes are written using JSON format, which is human understandable, light, and can be parsed by many systems.

To test the system, two different platforms were used: Keepon Pro [20] and an own made platform called Triskarino. Keepon has been used to test the interoperability of the system to different platforms, while Triskarino has been used in diverse case studies to explore emotion projection with a non anthropomorphic platform.

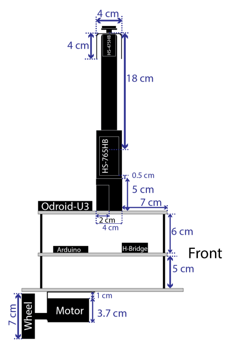
A. *Keepon Test*

To test the system with Keepon (Fig. 6), it was just necessary to implement the platform's controllers in each one of the simple action ROS nodes. Given that this platform does not have the capability to displace its body, the action "move body" was not implemented. Once added these controllers to the system, we proceeded to modify the configuration files to use this platform instead of Triskarino. With this small modification, we were able to change from one platform to other. In the video[1] is shown the test where Keepon had to move its torso forward and backward to express a "happy" emotion. The action is given by console telling the robot to bend the torso to a desired angle in x. The torso oscillation in y is added automatically by the system following the description defining how to express happiness.

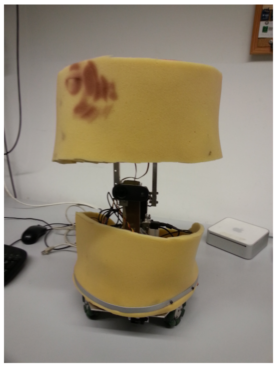
B. *Triskarino Test*

The system has been widely used with Triskarino (Fig. 7) during our studies on projecting emotions with a non human like platform [21]. In these studies, the system was used

[1]YouTube video name Emotional Enrichment System, url: https://youtu.be/bRSXQ0rzkO8



(a) Triskarino design.



(b) Triskarino with its cover.

Fig. 7: Triskarino platform.

in different experiments to move the robot on a straight line showing different emotions, each one selected from our command console. Moreover, to show the whole capabilities of the emotional enrichment system and verify if it could be used in real time, a little scene was set up. The scene is a modification of the first part of the balcony scene of Shakespeare's Romeo and Juliet play [19]. The simplified sketch of our version could be seen in Figure 1. As it can be seen the stage was divided in 81 squares and the positions were given to the system in terms of the desired square to be reached, which allows the robot to adjust to the stage's dimension. The whole sequence of actions were specified as unique action, having several sequential nodes and just move body action. The other actions, such as oscillate body or blend upper body were added online accordingly the desired emotion.

VII. Conclusions and Further work

An Emotional Enrichment System has been designed and implemented to enrich robots' movements with emotions. To achieve this, an Emotional Execution Tree created from three different types of nodes was used: simple actions, parallel, and sequential. Simple actions are functions that map a set of parameters to specific movements. Sequential nodes execute the sequence of actions associated to this type of node in a sequential order, while parallel executes them all at the same time. To enable synchronization among simple actions, parallel nodes could be one of four different subtypes, and sequential just one of two different subtypes. A formalization of the Emotional Execution Tree and the principal considerations during the implementation of the system have also been provided. To show the system's versatility, it was used with quite different platforms such as a Keepon Pro and Triskarino. Keepon was used to perform simple actions, while Triskarino was used to test complex actions with different parametrizations of emotions. The system has been used in different case studies, to implement emotion descriptions.

The obtained results show that the system could enrich the robotic actions with emotions, which could be parametrized from configuration files. Additionally, the design of the system makes it possible to adopt it for different platforms using the same action description.

References

[1] B. Reeves and C. Nass, *The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places*. New York, NY, USA: Cambridge University Press, 1996.

[2] A. Paiva, I. Leite, and T. Ribeiro", ""emotion modeling for social robots"." [Online]. Available: "//www.oxfordhandbooks.com/10.1093/oxfordhb/9780199942237.001.0001/oxfordhb-9780199942237-e-029"

[3] C. Breazeal, *Designing Sociable Robots*. Cambridge, MA, USA: MIT Press, 2002.

[4] S. Embgen, M. Luber, C. Becker-Asano, M. Ragni, V. Evers, and K. O. Arras, "Robot-specific social cues in emotional body language," in *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN'12)*. USA: IEEE Computer Society, September 2012, pp. 1019–1025.

[5] J. Li and M. H. Chignell, "Communication of emotion in social robots through simple head and arm movements," *I. J. Social Robotics*, vol. 3, no. 2, pp. 125–142, 2011.

[6] L. Brown and A. M. Howard, "Gestural behavioral implementation on a humanoid robotic platform for effective social interaction," in *Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, Aug 2014, pp. 471–476.

[7] M. Saerbeck and A. J. N. van Breemen, "Design guidelines and tools for creating believable motion for personal robots," in *RO-MAN*, 2007, pp. 386–391.

[8] A. Beck, A. Hiolle, A. Mazel, and L. Cañamero, "Interpretation of emotional body language displayed by robots," in *3rd ACM Workshop on Affective Interaction in Natural Environments*, 2010, pp. 37–42.

[9] P. Ekman, *Emotions Revealed : Recognizing Faces and Feelings to Improve Communication and Emotional Life*. Owl Books, Mar. 2004.

[10] C. Breazeal, A. Brooks, J. Gray, M. Hancher, C. Kidd, J. McBean, D. Stiehl, and J. Strickon, "Interactive robot theatre," in *Proceedings of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems(IROS 2003)*, vol. 4, 2003, pp. 3648–3655.

[11] H. Knight, S. Satkin, V. Ramakrishna, and S. Divvala, "A savvy robot standup comic: Online learning through audience tracking," TEI 2011, January 2011.

[12] G. Hoffman, R. Kubat, and C. Breazeal, "A hybrid control system for puppeteering a live robotic stage actor," in *RO-MAN 2008*, M. Buss and K. Kühnlenz, Eds. IEEE, 2008, pp. 354–359.

[13] LAAS-CNRS, "Roboscopie, the robot takes the stage!" http://www.openrobots.org/wiki/roboscopie.

[14] C.-Y. Lin, C.-K. Tseng, W.-C. Teng, W. chen Lee, C.-H. Kuo, H.-Y. Gu, K.-L. Chung, and C.-S. Fahn, "The realization of robot theater: Humanoid robots and theatric performance," in *International Conference on Advanced Robotics, 2009. ICAR 2009.*, 2009.

[15] R. Murphy, D. Shell, A. Guerin, B. Duncan, B. Fine, K. Pratt, and T. Zourntos, "A midsummer night's dream (with flying robots)," *Autonomous Robots*, vol. 30, no. 2, pp. 143–156, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10514-010-9210-3

[16] S. Kopp, B. Krenn, S. Marsella, A. N. Marshall, C. Pelachaud, H. Pirker, K. R. Thórisson, and H. Vilhjálmsson, *Towards a Common Framework for Multimodal Generation: The Behavior Markup Language*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 205–217.

[17] H. Gunes, B. Schuller, M. Pantic, and R. Cowie, "Emotion representation, analysis and synthesis in continuous space: A survey," in *Face and Gesture 2011*, March 2011, pp. 827–834.

[18] J. M. Angel-Fernandez and A. Bonarini, "Robots showing emotions," *Interaction Studies*, vol. 17, no. 3, pp. 408–437, 2016.

[19] R. S. Company, "Royal shakespeare company - romeo & juliet, on stage footage - ny," http://www.youtube.com/watch?v=FHoaPLO6Zd8.

[20] H. Kozima, M. P. Michalowski, and C. Nakagawa, "Keepon: A playful robot for research, therapy, and entertainment," vol. 1, no. 1, January 2009.

[21] J. M. Angel-Fernandez and A. Bonarini, "Robots showing emotions," *Interaction Studies*, vol. 17, no. 3, pp. 408–437, 2017.