POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Design and implementation of a multi-modal framework for scenic actions classification in autonomous actor-robot theatre improvisations

Laurea Magistrale in Computer Science Engineering - Ingegneria Informatica

Author: Lorenzo Farinelli

Advisor: Prof. Andrea Bonarini

Co-advisor: Federico Espositi

Academic year: 2020-2021

1. Introduction

Robots (and AI-powered machines in general) are seeping into the fabric of modern society, assisting or even replacing humans in a variety of contexts. Besides being employed as industrial tools, a growing body of research and commercial applications is envisioning robotic social companions for everyday life.
The goal of a robotic social companion able to interact with humans in *unconstrained* settings is, however, still far from being achieved, due to both technical and ontological problems. For this purpose the art of *theatre* is often employed as a testing ground, being it a representation and simulation of reality, and in particular of social interactions. Several works have been developed to create robotic actors, for which *Lu* proposed a categorization into 9 classes of increasing complexity [1]. Developing a robot occupying the last two classes, characterized by complete autonomy and expressiveness, is a challenging task. The work presented here aims to push research in this direction by proposing a framework for an autonomous robot improviser. The robot collects data from an actor and detects performed scenic actions, enabling the selection of a proper reply to make the scene continue

2. Goal and methodology

Given the difficulty of developing a robot able to act in unconstrained situations, the problem has been decomposed into smaller and simpler parts. The following simplifying assumptions have in particular been made:

- the stage is empty and of fixed size
- only a single actor is present in the stage
- interaction between the robot and the actor is modeled as a turn-based communication

Regarding this last point, a simplified scheme of human communication has been adopted. At the beginning the attitude of the interlocutor is assessed in an unconscious and instantaneous way, based on non-verbal signals and classified into a defined set of possibilities. This classification produces an unconscious and subjective emotional response depending on the current *emotional state* and *personality*, which triggers a response that could be moderated by the *rational mind*. This reaction is in turn processed by the interlocutor, giving rise to a chain of interaction loops.

2.1. Reference framework

To navigate the complexity and lack of quantitative rules of improvisation, a *reference frame-*

work has been developed, to outline the relevant input features and scenic actions to be detected by the system. Using sources and theories of different kinds and fields several dimensions have been identified, whose implementation is described in the next sections.
For the same reasons of simplicity explained at the beginning of the section this first version of the framework is based only on visual and spatial perceptions, to reach a stable and functional implementation on which other more complex and expressive semantic layers can be placed.

2.2. Evaluation

Evaluation of the implemented system has been conducted in two ways, to either assess the computational performance and correctness of implemented algorithms and to observe their behavior in *"in-the-wild"* interactions with humans. For this purpose a small square stage of 4 meters each side has been set up inside AIRLab, to provide a controlled testing environment without external disturbances.

3. Relevant inputs and outputs

The goal of the implemented system is to make a robot *"understand what is happening"* on the stage and for this purpose the concept of *scenic actions* has been defined. A scenic action is a combination of input features that carries a specific and self-contained meaning that timely characterizes the context of the scene and the actor's feelings and emotions.
Recognition of these actions is what gives the robot autonomy on stage and their definition and implementation have been crucial tasks in the process. It has been in fact necessary not only to understand which features are relevant for humans to characterize actions in an improv scene, but also which can be computationally retrieved and processed.
In the following all the features used by the system as inputs or intermediate results are described, showing their origin and motivation as well as methods for their retrieval and processing. These features are based on the following considerations:

- the meaning of someone else's actions, and the underlying intention, can be inferred from their expressed emotion
- emotions play a fundamental role in disambiguating possible interpretations of the same interaction
- emotions are usually expressed through facial expressions, bodily movements and spatial relationships between characterizes
- movement features (e.g. speed, extension, direction) externalize the emotional state of a character
- the distance and eye contact between two characters characterize their levels of intimacy and trust
- the above expressive media are sufficient to understand the unfolding of a story, as in the case of silent movies and mimic sketches

As a result, *emotion*, *eye contact* and *proxemics* have been identified as the key features to characterize the actor's performance.

3.1. Emotion

Several theories and computational models of emotion exist in the literature (with no general consensus on a single one) [2–6]. In this work the most popular among them has been implemented, namely the *discrete model* by *Ekman* [2], describing six universal emotions[1] linked to facial expressions. In the implemented system the actor's emotion is inferred from their facial expressions and body gestures using two neural networks sharing the same output space *(i.e. the six Ekman emotions plus a neutral state)*.

3.1.1 Emotion from face

Given the abundance of existing solutions and research projects in the field, this module has been implemented using *open-source libraries*. The PAZ Python library [7] has in particular been selected against other competitors such as DeepFace [8], on the basis of declared performance metrics. The library implements a MiniXception emotion classifier and offers perception and processing modules at different levels of abstraction. In this context, the network outputs a probability distribution over the seven emotion classes.

3.1.2 Emotion from body

Recognition of emotion conveyed through body movements does not instead feature this same

[1] anger, disgust, fear, happiness, sadness, surprise

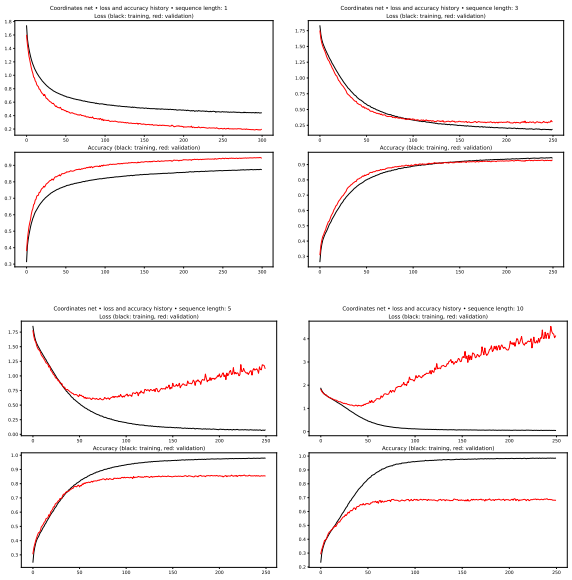


Figure 1: Plots of loss and accuracy of the network for different lengths of sequences

abundance [9], and hence a *custom neural network* has been designed and trained on a dataset of motion-capture data sequences of actors expressing emotions [10].
The implemented network is the result of several experiments on both input representations, model architectures and sequences length, aiming at the best accuracy. Tested solutions have in particular been a *graph network*, representing skeleton coordinates as a graph, and several variations of a *recurrent network* having as input sequences of descriptors computed from body landmarks using either spatial/kinematic information or with formulas inspired by the Laban Movement Analysis framework [11].
The chosen model is a recurrent network employing an LSTM and 3 fully-connected layers to classify sequences of body landmarks' coordinates (represented on a bidimensional plane centered in the hips) into a probability distribution over the seven emotion classes. The length of these sequences has been set to 3 poses after testing the training behaviour and *"in-the-wild"* performance on several values, as reported in figure 1. The network eventually reaches an accuracy of 0.93 on the validation set.

Pose estimation Since the network relies on body landmarks' coordinates, a pose estimation module has been added to the system to retrieve the location of body joints in the aforementioned coordinate system. Model selection has been guided by the parameters of latency and stability of predictions over time, to avoid noise due to the jittering of estimated joints. In the end, the chosen solution is MediaPipe [12], a set of open-source machine learning solutions developed by Google, targeted to real-time and live media use cases, and optimized for deployment on resource-constrained mobile devices. The output of the model has been reduced to a set of 13 landmarks (those used by the body emotion network) to which a normalization and coordinate change is applied, to make poses invariant to scale and consistent with the training dataset.

3.1.3 Emotion fusion

From these two sources, a final single emotion $\hat{E}$ is computed, to provide a "global" characterization taking into account all the ways in which the actor can express emotion. The fusion point is placed at the end of the two inference paths, being it the common approach especially when dealing with gestures [13]. Choice of the fusion method is based on researches and empirical observations that body gestures and facial expressions may sometimes portray different (and even contrasting) emotions, but with the latter being a more reliable and informative source. For this reason the final emotion is computed as a weighted sum, whose weights are inspired by the proposal in [14] ($\omega_{face} = 0.7, \omega_{body} = 0.3$): $\hat{E} = argmax\{E_{face} \cdot \omega_{face} + E_{body} \cdot \omega_{body}\}$

3.2. Eye contact

Eye contact (*gazing*) is one of the fundamental cues of non-verbal communication and helps to characterize different situations, understanding for instance whether a certain utterance is directed or not to the robot and whether the actor wants it to be hidden or shown. Recognition of eye contact presence is implemented through the neural network proposed in [15], achieving an accuracy comparable to human experts, which yields the probability of eye-contact presence in a given face image patch.

3.3. Face detection

Both face-based emotion and eye-contact recognition models use a face image patch as in-

put. For this reason, face detection and cropping is shared and implemented just once. The OpenCV Haar Cascade detector has been chosen among available techniques, after comparing their trade-offs in terms of latency and accuracy. In particular, to avoid issues with tilted head poses, detection is repeated, in case of failure, over different rotations of the image.

3.4. Proxemics

Proxemics is a subfield of the studies on non-verbal languages concerned with the use and influence of space in human behaviour and communication. Space can in fact used and shaped by the actor to convey certain narratives and express their appraisal and attitude towards the robot. Five proxemics descriptors are computed in the system, from the knowledge of both the robot's and actor's locations at each time step (provided by third-party black-box systems).

3.4.1 Proxemic zone

The concept of proxemic zone has been introduced in [16] to describe discrete social zones surrounding an individual, each linked to a certain degree of familiarity of people let inside. Inspired by this, three zones have been defined to characterize the relationship with the actor according to their distance from the robot, namely *intimate, neutral* and *no-interaction*.

3.4.2 Movement direction and speed

Distance between the actor and the robot is also considered in its temporal evolution, to understand if they are getting closer or apart. In particular, the system computes the movement's direction[2], which can describe their interest in the interaction and their trust for the other, and speed[3], which denotes the urgency of underlying intentions. Speed is in particular computed from polar coordinates both on the distance segment and on the corresponding circle, to describe frontal and lateral movements.

3.4.3 Relative and absolute position

In addition to the zone, the system computes a discrete classification of the actor's position around the robot, to interpret different nuances of the same proximity. One of the four cardinal directions is given in output, based on the θ angle of the actor's pose in polar coordinates. Besides this, the robot's and actor's locations are characterized w.r.t. to the audience into nine stage positions using the *Stage Grid* model.

[2]approach, retreat, stillness
[3]slow, medium, fast

Emotion	Direction	Speed	Zone	Scenic Action
Anger	appr	fast	int	Attack
	appr	slow/med	int	Intimidation
	still	-	neut	Scolding
	retr	any	no int	Holding grudge
Disgust	retr/still	any	neut/no int	Refuse
	appr	any	any	Perplexity
Fear	appr/still	fast / medium	int/neut	Share fear
	appr	slow	int/neut	Caution
	still	-	neut/no int	Hesitancy
	retr	slow	any	Shock
	retr	medium/fast	any	Escape
Happiness	-	-	int	Share joy
	-	-	neut	Greet
	-	-	no int	Happy person
	retr	any	neut/no int	Satisfaction
Sadness	appr	any	int	Share sadness
	-	-	neut/no int	Sad person
	retr	any	neut/no int	Disappointment
Surprise	appr/still	any	int/neut	Share surprise
	still	-	no int	Astonishment
	retr	any	neut/no int	Disbelief
neut	any	any	any	neut

Table 1: Scenic action recognition. The first four columns contain input values, while the last indicates the classified action. Feature values are expressed as high-level labels that are numerically grounded using fuzzy sets.

3.5. Scenic actions classification

Scenic actions summarize all the features described so far into a representation of the actor's behaviour. Their classification has been defined following the methodology below:

- understanding what a scenic action is and how it can be characterized
- defining in detail the output space, listing the relevant actions to be detected
- defining a proper action representation
- surveying and choosing available classification strategies

In the end the classification in table 1 has been developed. It's important to point out that even though not all the described input features have been assimilated in this first classification framework, they are still available as outputs to be used in future or by third-party systems.
The problem of identifying a scenic action can be reduced to a multi-class classification problem based on the listed input features. For the sake of simplicity and functionality, a rule-based system has been chosen among the available methods, with fuzzy rules to smooth the classification output over time, making it more robust to fluc-

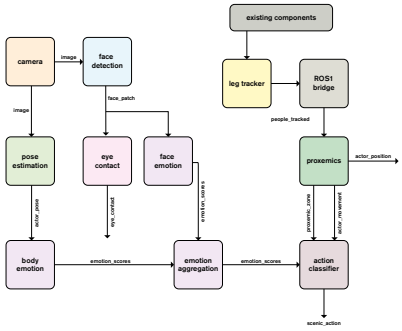


Figure 2: High-level block diagram of the system's architecture in the ROS environment

tuations in input values.

4. System deployment

4.1. Target robot

The implemented system has been eventually deployed on a physical robot, to also test its behaviour in a real-world setting. The target has been in particular Robocchio [17]: an omnidirectional 120 cm tall robot developed in AIRLab to perform script-based theatrical representations through body movements and eye expressions. For this reason, the robot already features several hardware and software components that have been reused for perception and proprioception. Two laser sensors are in particular present, which, through a proper merging technique, provide scans of the whole area surrounding the robot (with a radius of $4m$). This information is exploited by a third-party leg tracker [18] to estimate the location of the actor. Robocchio's control system is built using the ROS1 framework and has been extended with ROS2 nodes to implement the designed framework, using the ROS Bridge package for interoperability.

4.2. System architecture

The system architecture has been designed in a modular fashion, creating (where possible and not hindered by hardware and performance issues) a node for the processing of each input/output feature and a topic for its distribution to consumers inside and outside the system, as shown in figure 2. Modules exploiting image frames captured by the on-board camera have however been aggregated as separate threads in a single node. This is due to memory management issues in both ROS and Python, that add latency when accessing frames exchanged through conventional messages and noticeably slow down the inference models. All the other nodes trivially wrap the models and algorithms computing the data described in section 3, interfacing with the publish/subscribe paradigm of ROS for their exchange within the system.

4.2.1 Data collection

Physical deployment of the framework also brought about several issues to be addressed, mostly related to data acquisition and collection.

Windowing The robot senses the environment and actor in a continuous way, running at an average of 20FPS. Humans, on the other hand, understand others' expressions at lower frequencies and movements often make sense only when considering their temporal evolution. For this reason, a windowing mechanism has been implemented (employing sliding windows of around 1 second), to both smooth data along time and compute a more stable and robust characterization.

Handling of missing observations Given the context of the work, features listed as inputs may not always be available or detected, for artistic reasons, by the system (*e.g. the actor can be outside the camera's field of view*). In these cases different approaches are adopted: proxemics descriptors are computed by placing the actor at an imaginary point outside the stage in front of the robot, whereas for emotions and actions a neutral fallback classification is given.

Handling of robot movements The quality of collected data can also be spoiled by movements of the robot's body, introducing noise (*e.g. wobbling of camera*) or causing misinterpretation (*e.g. actor that seems to be approaching when it's actually the robot that is moving*). For this reason, the system is subscribed to a feedback signal from actuator systems, to ignore incoming inputs when the robot is moving and to collect them again once it has finished.

5. Conclusions and future steps

What has been described in this thesis is the design and implementation of a framework to ren-

der a robot able to participate in an improvised exhibition with a human actor moving on the stage, identifying their scenic actions through a set of sensory observations.
This first implementation of the framework is based on some simplifying assumptions and takes into account a reduced set of expressive elements, which are however sufficient to characterize the actor's intentions and enable a chain of scenic actions to develop an improv scene. This work has in fact focused on laying out a foundational architecture, to both provide a stable ground for the implementation of the robot's sensing capabilities and validate the feasibility of the designed framework in the face of computational and real-time constraints. Moreover, the availability of a physical system can help a faster improvement of the framework's expressiveness, by means of direct interactive tests.
This implementation can therefore be considered as the first block of a cognitive pipeline for an autonomous robot improviser and can find its place even in the more general field of HRi[4], to improve the situational awareness of robots and create more engaging and effective experiences.

5.1. Future directions

During brainstorming sessions several interesting ideas emerged; even though unrealized, they are still worth to be mentioned to highlight possible future developments of this work.
The framework can in fact be extended with additional semantic layers and inputs, adding for instance a verbal and non-verbal characterization of the actor's speech. Besides sheer sensory data, the system can be also made aware of meta-features describing the quality and dramatic evolution of the scene, to guide the selection of proper replies and make the story develop or come to an end. Among these meta-features it could be interesting to implement the concept of entropy, to represent how each action and scene configuration contributes to the creation of possible interpretations and plot lines. This can be moreover coupled with a formal representation of narrative archetypes through modal or descriptive logics, to enable reasoning and planning. Eventually, a dataset is proposed to be built, annotating sequences of robot's observations with the corresponding scenic actions, to enable training of more complex classification models or even a neural synthesis of the robot's replies, removing the human element from its behaviour definition and thus getting closer to the goal of full autonomy.

[4]Human-Robot-interaction

References

[1] D. V. Lu, "Ontology of robot theatre," in *Proc. Workshop Robotics and Performing Arts: Reciprocal Influences, ICRA*, 2012.

[2] P. Ekman, "An argument for basic emotions," *Cognition and Emotion*, vol. 6, pp. 169–200, 5 1992.

[3] J. A. Russell, "A circumplex model of affect.," *Journal of Personality and Social Psychology*, vol. 39, pp. 1161–1178, 1980.

[4] K. R. Scherer, "Appraisal theory.," 1999.

[5] R. PLUTCHIK, "Chapter 1 - a general psychoevolutionary theory of emotion," in *Theories of Emotion* (R. Plutchik and H. Kellerman, eds.), pp. 3–33, Academic Press, 1980.

[6] S. Tomkins, *Affect imagery consciousness: Volume I: The positive affects*. Springer publishing company, 1962.

[7] O. Arriaga, M. Valdenegro-Toro, M. Muthuraja, S. Devaramani, and F. Kirchner, "Perception for autonomous systems (paz)," 2020.

[8] S. I. Serengil and A. Ozpinar, "Lightface: A hybrid deep face recognition framework," in *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pp. 1–5, 2020.

[9] B. De Gelder, "Why bodies? twelve reasons for including bodily expressions in affective neuroscience," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 364, no. 1535, pp. 3475–3484, 2009.

[10] M. Zhang, L. Yu, K. Zhang, B. Du, B. Zhan, S. Chen, X. Jiang, S. Guo, J. Zhao, Y. Wang, *et al.*, "Kinematic dataset of actors expressing emotions," *Scientific data*, vol. 7, no. 1, pp. 1–8, 2020.

[11] C. Larboulette and S. Gibet, "A review of computable expressive descriptors of human motion," in *Proceedings of the 2nd International Workshop on Movement and Computing*, pp. 21–28, 2015.

[12] C. Lugaresi, J. Tang, H. Nash, C. Mcclanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, M. Grundmann, and G. Research, "Mediapipe: A framework for building perception pipelines," 6 2019.

[13] L. Wu, S. L. Oviatt, and P. R. Cohen, "Multimodal integration-a statistical view," *IEEE Transactions on Multimedia*, vol. 1, no. 4, pp. 334–341, 1999.

[14] M. J. Gielniak and A. L. Thomaz, "Enhancing interaction through exaggerated motion synthesis," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pp. 375–382, 2012.

[15] E. Chong, E. Clark-Whitney, A. Southerland, E. Stubbs, C. Miller, E. L. Ajodan, M. R. Silverman, C. Lord, A. Rozga, R. M. Jones, *et al.*, "Detection of eye contact with deep neural networks is as accurate as human experts," *Nature communications*, vol. 11, no. 1, pp. 1–10, 2020.

[16] E. T. Hall, R. L. Birdwhistell, B. Bock, P. Bohannan, A. R. Diebold Jr, M. Durbin, M. S. Edmonson, J. Fischer, D. Hymes, S. T. Kimball, *et al.*, "Proxemics [and comments and replies]," *Current anthropology*, vol. 9, no. 2/3, pp. 83–108, 1968.

[17] L. Bonetti, "Design and implementation of an actor robot for a theatrical play," 2021.

[18] A. Leigh, J. Pineau, N. Olmedo, and H. Zhang, "Person tracking and following with 2d laser scanners," in *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 726–733, IEEE, 2015.

POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Design and implementation of a multi-modal framework for scenic actions classification in autonomous actor-robot theatre improvisations

Tesi di Laurea Magistrale in
Computer Science Engineering - Ingegneria informatica

Author: **Lorenzo Farinelli**

Student ID: 939565
Advisor: Prof. Andrea Bonarini
Co-advisors: Federico Espositi
Academic Year: 2020-21

Abstract

Social robotics aims to create autonomous robotic companions assisting humans both in work and leisure time, whose interaction mechanisms highly resemble those employed in interpersonal communication. Such goal is however far from being achieved, and no robot has been yet developed able to interact with humans in unconstrained settings.

In this context the art of theatre can come into help, as an ideal workshop where to develop, in a controlled environment, the social aspects of the interaction. Several works in human-robot interaction have in fact adopted theatre as a training field, motivated by its ability to simulate reality and replicate structures of human communication.

The work presented here follows this tradition as well, giving its contribution to novel forms of improvisation between human actors and robots. What is going to be proposed is in particular a framework to render an autonomous robot able to perceive the emotional state and intentions of the actor, detecting the scenic actions they perform on stage.

Based on theories from neuroscience, psychology, acting and choreography, the framework acquires, processes and analyses in real-time several features of the actor's performance, through a set of sensors mounted on the robot and computational algorithms.

From collected features a classification into a discrete set of basic performative actions is derived, which timely characterises the actor's attitude and intentions. This output can be subsequently attached as an input to third-party systems, to produce for instance a physical reply. In this way it will be possible to design cognitive pipelines of diverse complexity, adding modules related to the robot's psychology, mobility and aesthetics.

To ensure a functional and optimal behaviour, the implemented system has been designed using a mixture of both third-party open-source state-of-the-art algorithms and manually trained neural networks, choosing among the available solutions on the basis of several trade-offs, which are documented and compared in the document.

Keywords: Autonomous Robots, Theatre, Improvisation, Emotion, Human-machine interaction

Abstract in lingua italiana

La robotica sociale ambisce a creare robot autonomi che interagiscano con le persone replicando i meccanismi propri della comunicazione umana. L'obiettivo è quello di integrarsi con i loro stati emotivi, reagendo di conseguenza per poterle aiutare (in diverse attività) nella maniera più naturale possibile. Quest'obiettivo rimane tuttavia ancora lontano, e nessun robot finora sviluppato è in grado di interagire con le persone in situazioni arbitrarie. In questo contesto è di grande aiuto l'arte del teatro, che si configura come un terreno ideale in cui testare gli aspetti sociali dell'interazione uomo macchina grazie alla sua capacità di simulare la realtà e le strutture comunicative del linguaggio.

Diversi robot attori sono stati infatti sviluppati negli ultimi anni e il lavoro qui presentato si inserisce in questa linea di ricerca con l'obiettivo di creare nuove forme di improvvisazione teatrale tra robot e umani.

Viene infatti proposto un framework per realizzare un robot autonomo capace di cogliere lo stato emotivo e le intenzioni dell'attore, identificando le azioni sceniche compiute sul palco. Questo framework acquisisce ed elabora in tempo reale dati quantitativi e qualitativi sulla performance dell'attore, attraverso opportuni sensori collegati al robot e metodi computazionali ispirati da teorie e modelli ripresi da neuroscienza, psicologia e coreografia, Da tali dati vengono classificate delle azioni performative di base, che caratterizzano in maniera puntuale l'attitudine e le intenzioni dell'attore nei confronti del robot.

Questo output può essere inoltre usato da sistemi terzi, per realizzare una risposta fisica che faccia proseguire la scena. In questo modo è possibile realizzare pipeline cognitive di varia complessità ed espressività, andando a definire moduli legati agli aspetti psicologici, motori ed estetici del robot.

Per garantire un funzionamento ottimale il sistema è stato progettato utilizzando sia tecnologie open-source sia implementando nuovi modelli come reti neurali, selezionando le opportune soluzioni attraverso un'analisi costi-benefici descritta nel corso del documento.

Parole chiave: Robot autonomi, Teatro, Improvvisazione, Emozioni, Interazione uomo-macchina

Contents

Introduction

As you set out for Ithaka
hope your road is a long one,
full of adventure, full of discovery [...]
May there be many summer mornings when,
with what pleasure, what joy,
you enter harbors you're seeing for the first time

Ithaka - Constantine Cavafy

Robots (and AI-powered machines in general) are nowadays seeping into the fabric of modern society, assisting or even replacing humans in a variety of contexts. Not only they are employed as industrial tools, taking over heavy, repetitive, or precision tasks, but there is also a growing body of research and commercial applications envisioning robotic social companions for everyday life, for instance replacing caregivers [105] in assisting elderly people or children affected by autism spectrum disorder.

An all-round robotic social companion, able to interact with humans in *unconstrained* settings, however, is a goal still far from being achieved. *Breazeal* claimed in [18] that robots "*treat us either as other objects in the environment, or at best they interact with us in a manner characteristic of socially impaired people*"; this observation is still valid nowadays, not only for the technical issues related to formalizing and coding human social norms into a machine but also (and maybe most importantly) for an additional ontological dimension. As in fact described in [68], robots that socially interact with humans face several main problems, such as:

- different expressive media w.r.t. humans (and hence different degrees of freedom) that *"hinders the ability to mimic common physical human cues"* and makes the robot being *"perceived as unresponsive or unintelligent"*
- robot's intentions are not always clear or understandable, and there is a tension between simple movements, perceived as ambiguous, and complex motions failing to fully convey the intended information

- human motions should not only be observed and recognized in a computational manner, but should also be framed into a larger context to make sense of the current situation in which the interaction is taking place. This, to be achieved, "*requires a sophisticated model of human behaviour*" that can also take into account cultural differences.
- evaluation of human-robot interactions is often based on the subjective experience of testers, and too dependent on the "*unpredictability of the human element*"

For these reasons a proper conceptual environment is needed, in which to design, experiment, and validate different paradigms and capabilities of sociality and human-robot interaction. A playground that *resembles reality* yet at the same time is controlled enough to introduce simplifications in the acquisition and processing of external inputs with respect to the complexity of the real world outside.

An environment of this kind can be found in the world of theatre, which has been since its birth a way to represent reality and decompose it into smaller pieces to fathom the depths of the human condition, the mysteries of life, and the collective consciousness and rituals of society. The theatre stage is in fact a space and time outside the ordinary space-time, in which common rules can and cannot apply and new ones can be defined; thanks to this, theatre has the power to be the ideal workshop and test-bench in which different possibilities can be brought to life, simulated and assessed (being them new ideas, technologies, events...) in a collective ritual to isolate the alien element and look at it through different lenses, hypothesizing the impact of its assimilation or rejection.

> *If a robot can be programmed to convey its intentions on stage, then we can be sure that the robot is at least conveying some of the right social cues, which can then be transferred to more traditional human-robot social situations*
>
> D. Lu - Human-robot interactions as theatre [68]

State of the art

All the considerations made so far find evidence and validation in the existence of several works at the intersection of robotics and theatre, which are briefly described in this section to provide a general overview about the state of the art.

Different approaches have been followed so far in the development of robot actors, and a possible categorization has been proposed in [67] based on the two orthogonal axes of *autonomy* and *control*.

Autonomy describes the complexity of algorithms used to operate the robot and highlights which entity is actually selecting the action to perform.

- *Human*: actions are decided and produced by a human operator, either in the forms of remote control, hardcoded motions, or generative mapping of human movements *(e.g. motion capture)*
- *Algorithmic*: the robot is fully autonomous and its behaviour is determined by the results of an algorithm, without human intervention
- *Hybrid*: the robot's behaviour is a blend of the two, with a human adjusting the result of an algorithm or vice-versa

Control provides instead a classification of how awareness of the surrounding environment (if present) is implemented and how the robot's behaviour is adjusted over time.

- *open-loop*: no adjustment is made to the robot's behaviour, that instead *"does the same exact thing in every performance regardless of external factors"*
- *closed-loop*: a feedback mechanism is present, which *"allows the robot to react to the given performances"*
- *free*: no constraints are put on the robot, that instead *"can do nearly anything"*

Combination of values on these two axes eventually produce nine possible classes of increasing complexity in which a robot actor can fall, that can be further grouped into four categories as shown in figure 1.

Many of existing examples of robot actors use the audience as a feedback signal to learn or adjust their behaviour. This is the case for Data, a stand-up robot comedian able to select jokes according to the reactions of the public [58], whereas in ComedyLab [57] a humanoid robot performs a script by analyzing spectators' faces and addressing specific people. A similar approach is employed in [19], where a serpentine anemone-like creature (called *Public Anemone*) interacts with the crowd, whose inputs can elicit different responses, such as curiousity or fear. The behaviour of the robot is then based on a joint combination of both the people's stimuli and its internal drives, creating a tension between responding to people (wich are encouraged to compete for the robot's attention) and fullfiling daily chores *(e.g. watering plants, drinking, bathing)*. Relationship with the audience is also the subject of a more recent work [108], where its response to a robot comedian has been analysed through multiple performances, varying the robot's timing and adaptivity to assess variations in engagement and perceived humor.

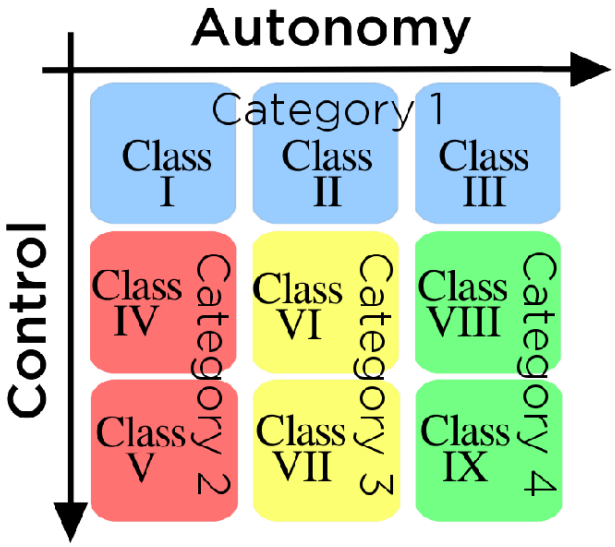


Figure 1: Diagram showing the classification in classes and categories of robotic actors proposed by *Lu* [67]

The robot developed in [71] uses instead the audience as the actual control signal, providing a mobile interface in which spectators are prompted to select a reply line and a gesture that will be uttered by the robot, namely a mechanical arm named *Pokey*. In this context the responsibility of carrying on the improvisation is completely placed in the hands of the human peer performer, which has to reply to Pokey's props in a meaningful way linking at the same time all the pieces of the story developed so far.

Another approach followed in the development of robotic actors is to have a predefined set of behaviours and lines, which are however selected in a non-deterministic and dynamic way to follow the evolution of a dramatic structure. The architecture proposed in [20] is in fact based on the activation of behaviours according to the robot's inner states, goals and other actors' utterances. It's also of interest the approach followed here to represent (and subsequently handle) the concept of an internal emotional state: instead of explicitly representing emotions and modeling how they work they are implemented in the form of *inner obstacles*, which can *"inhibit certain behaviours and encourage others"*. These obstacles are also characterised by an equilibrium value to which they decay over time (inspired by the work on emotional models of *Velasquez* [107]). By specifying different equilibria is therefore possible to realize different agents with different personalities all based on the same architecture.

Other approaches have then used humanoid robots, either in big scale such as *RobotThespian* or small in the case of *Nao*. Approaches of this kind, however, have suffered from mobility capacities, both for logistic and artistic reasons, making them to fully explore

the motion and spatial possibilities. Expressive motion has been considered in [84], where a robot is employed in a basic theatre improvisation game, and in [4], where a robot improvises a piece of dance by means of pre-defined. Both approaches, however, fail to obtain a fully autonomous performance and could be only tested only in a Wizard-of-Oz setting (namely being the robots assisted and controlled by a human operator).

Besides the development of acting and expressive capabilities in robots, other works focused on making their performance fluent and resilient to unexpected failures on stage, which can have a negative impact on the perception of the quality and of the robot itself. [35] studied in fact the development and exhibition of two robot performances, observing that failures occurred (in different forms) nearly in every day of rehearsal and during live shows. From this analysis the paper suggests four different strategies to face failures at different stages of the performance's development, characterised by different severity and required effort:

- *Showstoppers*: requiring a significant time investment to add new capabilities to the robot or to rewrite some sequences of behaviour
- *Stop-fix-try again*: addressing minor issues that can be fixed and tested on the fly in a few minutes (*e.g. adjusting robot's position, recharging batteries...*)
- *Co-performer accomodation*: leveraging the human peer to handle variations in the robot's performance and give them a coherent meaning
- *Human-centric replacement*: addressing *"catastrophic robot failures"* with a pre-planned and practised backup plan, such as remote-control or replacement of the robot's performance with a human

Scope of the thesis

Considering the classification of robot actors reported in figure 1, this work aims to give its own contribution to the fourth category, which envisions robots able to autonomously decide what to perform according to external circumstances.

The work presented in this document focuses in particular on a specific form of theatre: improvisation, inspired by the claim of *Breazeal* that *"introducing improvisation [...] makes the situation more unpredictable and less constrained, approaching open-ended interaction with people"*, enabling the robot to *"act/react in a convincing and compelling manner to the performance as it unfolds"* [19]. The goal is to bring a contribution to the development of autonomous robotic improvisers by designing and implementing a system that provides them with the needed perceptual and awareness capabilities, to make them

act in a fully functional and really autonomous way.

In theatrical improvisation, commonly referred to as *Improv* [55], actors do not have a fixed script, but instead dynamically create plot-lines on the stage based on props from the public or peer improvisers. Besides the ability of performers, improv is based on simple rules, such as *YES, AND...* [87], which prompts actors to accept proposals of the partners and reply back with an offer to make the scene go on. The success of an improv scene is usually determined by the reciprocal engagement of the actors.

From these basic rules and from the analysis of several kinds of materials, a framework for autonomous robot improvisations has been first designed and subsequently implemented on a physical robot. In this first implementation the framework is based on the following simplifications:

- the improvising robot has just a **single** human partner
- **no props** are used in the improvisation
- no speech is present, the only expressive capabilities that can be used (for both performers) are those offered by body movements, facial expressions and spatial relationships

and enbles the robot to classify the *scenic action* the actor is performing on stage, by interpreting data collected by on-board sensors. The resulting classified action is eventually made available as an output to other third-party systems, to make them select a scenic action to perform in reply.

Structure of the document

After a short background about robots and theatre, chapter 1 presents in detail the goal of this thesis, analyzing the context of theatre improvisation and listing the requirements and boundaries of the work, to eventually introduce the developed reference framework along with its evaluation procedures. Chapter 2 is concerned about the input features that have been found to be of relevance in achieving the goal, describing for each the underlying motivations and available theoretical models and algorithms for their extraction, recognition or processing. Outputs of the system derived from these inputs are instead the subject of chapter 3, which first explains the methodology followed in the identification of relevant scenic actions, to later list them and compare possible theoretical models that can be used for their computation. Before moving to the implementation details, chapter 4 introduces the physical robot actor on which the system has been deployed, analyzing its existing features and components and explaining those that had been added anew, both

from the hardware and software points of view. Chapter 5 provides all the details about how the system has been physically implemented to achieve the goal. The discourse has been organized into sections, with a one to one mapping to inputs and outputs listed in chapters 2 and 3, where the ratio followed in choosing a particular implementation is explained: for each variable are compared multiple solutions for its computation, and motivations for the choice of the implemented one are given. Chapter 6 eventually closes the document, summing up the work described until that point and framing it into the larger context of existing research in human-robot interaction, listing potential points of improvement and proposing several directions in which the system can grow.

1 | Goal and context

Given this preliminary introduction, setting the background and motivation for the exploration of novel forms of theatre in which human and robotic actors cooperate in building and narrating a story, the document now focuses on the technical aspects related to the implementation of such robotic agents.

After an initial high-level description of the general goals and requirements, an analysis of the underlying conceptual perception and action classification framework is given, to map in the next chapters all the theoretical concepts into the corresponding physical quantities observable or computable by the system.

1.1. Goal of the thesis

Building a robotic improv actor can be considered a hard problem [69] and a parallelism can be traced with the hard problem of consciousness. To create in fact an artificial agent able to interact in unconstrained situations with humans, such as the case of an improv scene, there could be two possible ways:

- mimicking the behavioural patterns usually found in human-human interaction (HHI), creating a human-like companion
- grounding the robot's behaviour on a newly defined behavioural scheme, possibly but not necessarily inspired by other lifeforms

In both cases is fundamental to have a preliminary understanding of how this patterns originate and evolve. This is necessary not only to replicate them in the robots' architecture and to guide their decisions and actions, but also to meet the expectations their increasing complexity and capabilities create.

Social models have in fact been shown to often come into play during all sorts of interactions, as a way to make sense of the world and its complexity and a tool to decode the behaviour of either living or inanimate entities [18, 31, 81]. Moreover, this understanding of communication mechanisms involved in HHI are thought to be the key to further

develop artificial intelligence and to *"achieve an effective human-robot collaboration"* [45].

In light of these remarks, it is necessary to split the problem into smaller parts, that could be solved independently or in a cascade fashion and eventually be integrated in a modular architecture of ever-growing complexity and expressiveness. This approach can be thought as inspired by the concept of modularity of mind [83] and Baar's *Global Workspace Theory* [9], which conceives the mind as composed of several specialized low-level modules that respond to inputs and pool their outputs in a central *"workspace"*, where they activate high-level modules using a matching mechanism, and consciousness becomes a spotlight that at each time brings certain processes to surface, leaving the others as unconscious.

In this way the work can be organized and carried out by listing different semiotic and semantic layers of increasing complexity and by developing the foundational architecture on which these layers can be incrementally applied.

A possible distinction into layers could be the following:

- **Spatial**, related to spatial and proxemics relationships
- **Visual**, related to visual cues and body language
- **Auditory**, related to acoustic features of the interaction (e.g. tone, rhythm)
- **Activity**, related to the understanding of mimed actions and their context
- **Linguistic**, related to the understanding of spoken sentences and their context

The goal of this thesis is therefore to design and implement the aforementioned architecture, as well as the first levels in the hierarchy, in a way that could enable an easy future implementation of additional inputs and internal processes to enable the other layers.

1.1.1. Boundaries

Before proceeding to describe how this goal has been achieved it is important to set the boundaries of the work, to identify the responsibilities of the architecture to be developed.

Interaction between human actors and their robotic peer can be modelled as a *turn-based* communication, whose structure and mechanisms can be grouped at a high level in three distinct cascading interacting systems, as depicted in figure 1.1.

At each turn all the sensory inputs coming from the actor (possibly integrated with observation from the environment) are collected through the *sensing* module, and after being filtered and pre-processed are passed to the *understanding* block, where they contribute to update the robot's world model and to characterise several features of the interac-

tion, which are eventually used by the *decision* module to select and physically utter an appropriate response.

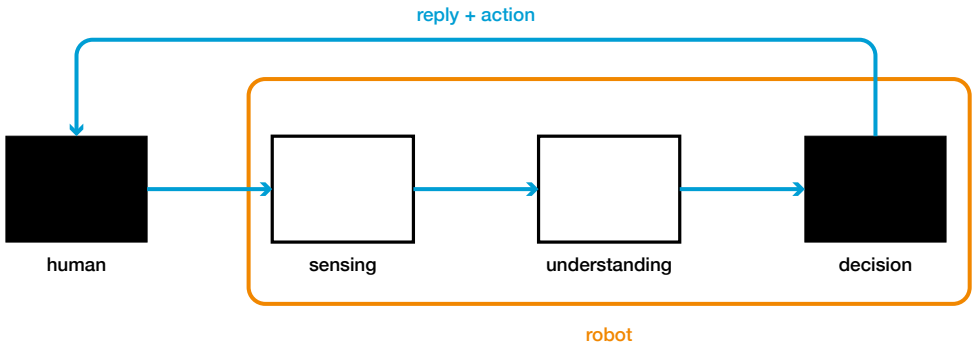


Figure 1.1: Block diagram showing at high level the interaction between a human actor and the architecture of the robot. Black-filled blocks represent parts of the system that are outside the scope of the work whereas white-filled blocks are the ones that will be designed and implemented.

In light of the remarks made at the beginning of this section, about the complexity involved in building an end-to-end architecture for a robotic improv actor, the work described in this thesis covers only the *sensing* and *understanding* blocks, as described in figure 1.1, in order to separate concerns and, by reducing the complexity, to focus only on the relevant problems of perception and awareness of the interaction. In particular, the work is based on the following assumptions:

- the robot acts in a fixed-size empty stage with no props
- the robot interacts with only a single actor present in the stage
- interaction between the robot and the actor is modelled as a turn-based communication, inspired by the human interaction model described in 1.1.1

To sum up, the work that is going to be presented in this document can be described as a multi-modal architecture for paired improv scenes between human and robotic actors, sensing spatial and visual communication cues for the classification and characterization of salient scenic actions during the interaction.

Human interaction model

The high-level turn-based description shown in figure 1.1 has then been enriched in the following simplified scheme of human interaction.

Its beginning is usually aimed at assessing the attitude of the interlocutor, which is mostly

done in an unconscious [22] and instantaneous way [112], based on non-verbal signals (movements, posture, face expressions, etc.) [73]. This assessment eventually classifies the attitude of the interlocutor within a defined set of possibilities (e.g., aggressive, amiable, cold, joyful) [36].

From this classification an emotional response arises, which is instantaneous, unconscious (*i.e. not driven by any rational reasoning*) and subjective (*i.e. the very same actions and expressions can trigger different emotional responses in different subjects*), depending in particular on the current *emotional state* and *personality*.

This emotional response causes (possibly) an alteration of the subject's current emotional state and triggers a response. In this moment the *rational mind* may intervene to moderate the reaction, driving it towards what is more opportune, socially acceptable, effective, etc., but still not producing a mediated answer, which instead comes later [106].

This reaction is in turn assessed and processed by the interlocutor, giving rise to a chain of interaction loops.

1.2. Analysis of the problem

In order to clearly understand what a robot improviser is supposed and required to do, it's important to point out the main differences that exist between improvisational and traditional/scripted theatre, to both set the boundaries of the problem space and suggest reasoning schemes that can be implemented.

Theatrical improvisation is in fact often considered, both by newbies and experienced actors, a "hard discipline". This is because it lacks a precise storyline and requires the actors to co-create a coherent and meaningful narration. To do so they all have to pay attention to a variety of semiotic elements: from the words uttered by their peers to the actions performed on stage, to the subtle changes in tone and facial expressions. In addition, besides looking inward to what happens on the stage, actors should also be aware of the "outside", to understand how their acting is perceived and to what extent it's engaging for the audience (even though in some improv performances there could be judges in the audience with a "boring" sign).

To sum up, what mainly characterises improvisational theatre is the higher number of degrees of freedom actors have in their acting. A freedom that is, however, a double-edged sword: if from one side it allows actors to have (almost) complete control over the unfolding of the story and shape its evolution as they like, on the other the fast decisions taken at each turn do not have the same quality guarantee of a well and long thought

script, where every line is carefully engineered to perfectly fit into the general canvas.

The absence of clear directions and guidance of a script can make actors feel left astray in the ever-growing forest of possible stories, and only few empirical rules exist to serve as a rudimentary compass, for instance:

- Yes, and..., often considered the foundation of improv, stating that improvisers should always accept offers (*endowments*) from other actors, and possibily use it to craft a new proposal in reply to add new elements to the story.
- Not asking open-ended questions: many times the success of improv scenes is based on ambiguity between the actors, generating many different interpretations that could lead to a variety of interesting stories. Asking an open-ended question forces the others to come up with just one, narrowing the possibilities of the interaction.
- Being open to change: (linked to the first rule) even if actors have a very interesting idea for their character, sometimes it's necessary for a good result to let go of it and embrace a more interesting proposal made by another peer.

These rules, however, as already mentioned, serve only as a very rudimentary compass to orient among the possible choices. Actually choosing which action to perform, and how, involves other different factors which are hard to track, because of their subjectivity and connection to one's personal experience, sensibility and references catalogue.

1.3. Reference framework

To shed some light on the complexity just described, and to help pointing out the main factors involved in perceiving and classifying an improv scene, a reference conceptual framework has been developed. This framework has a twofold goal: to identify which features generally contribute to scene awareness in human actors and to understand how they are processed and jointly considered to derive a classification of the ongoing interaction.

1.3.1. Methodology

In order to sketch the framework's architecture, it is necessary to understand not only which features are relevant and timely for humans during an improv session, but also which features can be computationally retrieved and manipulated to extract meaningful and valuable information, both from software and hardware points of view.

These two sets of features, however, are not one the subset of the other, since the use of a machine equipped with sensors can enable enhanced sensing capabilities not available

to humans *(e.g. detecting movement of a person behind one's back)*, and is therefore important to understand which of these perceptions can be useful during the interaction.

Moreover, as described in section 1.1, the goal of this thesis is to develop a **multi-modal** architecture for sensing theatrical improv scenes and classifying salient **scenic actions** during a human-robot interaction.

For these reasons the reference framework has been developed following two different and parallel approaches:

- **bottom-up**: from available relevant and measurable features understanding which actions and prototypical interactions can be detected
- **top-down**: from prototypical interactions and general scenic actions understanding which features are needed to be taken into account

This framework has been refined through several iterations, understanding which elements to add or remove by simulating the robot's behaviour with human testers.

In particular, this work has been carried out using different sources of inspiration:

- **short movies**, either acted or animated, to understand how an interaction is usually narrated and the most common features used to convey certain messages
- **hypothetical scripts** of an interaction between a human and a robot, to understand its possibilities and what is necessary for achieving them
- **improvisational scenes**, observed and analysed during improv theatre workshops[1], to understand recurring patterns and techniques that ensure a good result and engagement of the scene
- **direct simulation** of an improv scene, in which a person pretends to be the robot and acts accordingly, studying its limitations and needed capabilities
- **considerations on human communication**, to point out all the features that usually come into play (at different levels of subtleness) when characterizing the meaning and attitude of an interpersonal interaction

Results of this analysis are listed in the next chapter, where all the relevant variables measured or computed by the framework are described.

[1] thanks to the *Teatro delle Biglie* association

1.4. Experimental setup and evaluation

Evaluation of the project has been carried out in two different ways, touching the two different directions its goal suggests.

The developed system should in fact be able to sense the environment surrounding the robot, retrieving and processing data that could be useful to give a classification to actions performed by the human actor on stage. For this reason, the first point of evaluation is the correctness of the computational acquisition and elaboration of data, that can be carried out by means of unit testing, simulations and controlled input feeding.

However, given the context and scope of the project, which is strongly based on the interaction between the robot and a human actor, its evaluation cannot disregard a physical and interactive part, where the robot is actually observing a human actor performing on stage, to check how well models and algorithms developed in a sandbox environment perform "out in the wild". This kind of testing can in fact make several issues, unnoticed in the case of computational testing, come to the surface, such as:

- Non-idealities in the acquisition of data
- Physical components of the robot hindering the acquisition of data or introducing noise in the process *(e.g. visual occlusion)*
- Computational load of implemented models and algorithms, that introduce latency and prevent a real-time interactive behaviour of the system
- Different (lower) performance of machine-learning-based approaches on data captured in real-time with respect to their accuracy on validation datasets

For this reason, a small controlled stage has been set up in the AIRLab spaces, to have a stable environment in which the robot could localize itself and where interactive sessions could be carried out without external disturbances. The stage, shown in figure 1.2 is a $4m$ by $4m$ square, delimited by walls made of aluminium frames covered by white fabric, with an opening in the front to allow the presence of an audience like in a real theater stage.

This structure has been designed to be easily folded and unfolded as needed (reducing to a single flat wall) for two main reasons:

- minimizing the space occupied by the stage when the robot is not in use
- providing a portable structure that can be set up (virtually) in every space to make people interact with the robot

Figure 1.2: Stage set up inside AIRLab, composed of 3 walls made of aluminium frames covered by white fabric, in which evaluation sessions have been carried out

To help the robot localize itself in this stage, and provide specific landmarks in the map to disambiguate between the possible orientations, 3 objects have been put, asymmetrically, on 3 sides of the stage.

2 | Feature space

> ...I want to be suspended in the air above a flat stretch of open ground like the floor of an enormous arena. I would look down while I'm floating and there would be no horizon and no border."
> "An infinite development! Dysfunctional space!" She replied
>
> ROSA BARBA - A HOME FOR A UNIQUE INDIVIDUAL

This chapter describes all the features used by the system (either as inputs or intermediate results), showing their origin and motivation as well as methods with which they can be retrieved and processed.

The analysis of the sources listed in 1.3.1 led to the following considerations:

- the meaning of someone else's actions, and the underlying intention, is often inferred from their perceived emotions
- emotions play therefore a fundamental role in disambiguating possible interpretations of the same interaction
- emotions are usually expressed through facial expressions, bodily movements and spatial relationships between characters
- the above expressive media are by themselves sufficient to understand the unfolding of a story *(as in the case of silent movies and shorts)*
- the features of movements *(e.g. speed, extension, energy, ...)* can externalize the emotional state of a character
- the distance and eye contact between two characters can characterise the level of intimacy and trust between them

These have been subsequently mapped to established concepts in psychology, social sciences and choreography (described in the next sections) and summarized into a set of dimensions defining a feature space. By doing so the observations of the system at each time step can be described by a multi-dimensional feature vector.

2.1. Emotions

Emotions have been for centuries a topic of discussion among the scientific and philosophical communities, whose traces can be found even in ancient cultures [75], causing interdisciplinary research with the two-fold goal of enabling new applications and a better understanding of human emotional and cognitive processes.

This research is however far from convergence and no general consensus has been reached yet over the origin, structure and operational modes of emotions, due to several reasons:

- uncertainty about internal workings of the mind, related in particular to conscience and emotion formation
- inter-personal and inter-cultural differences in expressing, recognizing, and giving meaning to emotions
- influence of different psychological traditions

For this reason, several explanatory theories and computational models have been developed through history, marking each a mindset shift on how emotions and the inner psychological world is regarded in general.

These theories of emotions are often divided into three categories [74]:

- *Evolutionary* theories, focusing on a historical analysis of emotions to explain their presence and development
- *Social* theories, regarding emotions as a joint product of culture and society
- *Internal* theories, describing how emotions are internally generated by the self

Part of the difficulty in converging to a general theory of emotion is also due to the term itself, which encompasses a wide variety of physiological and psychological phenomena and therefore leads to ambiguity. Moreover, the term, going under the umbrella concept of *affect*, is often confused with its sibling *mood*. Mood is an affective state that happens at longer time scales, lasting even for hours, with higher stability (*the same mood can in fact contain a sequence of several emotions*) and subjectivity, whose triggers are harder to identify. In a nutshell, mood can de depicted as the canvas on which emotions originate or are projected.

2.1.1. Motivation

As described in [16], emotions enable the creation of connections between living beings, allowing them to understand what other individuals are experiencing or going through, by means of a comparison with one's own world model and past experiences that generates compassion and empathy, essential for the thriving of social structures.

> *Sento il tuo disordine*
> *e lo comparo al mio. C'è*
> *somiglianza. C'è lo stesso slabbro*
> *di ferite identiche. C'è tutta la voglia*
> *di un passo largo in una terra*
> *sgombra che non troviamo.*
>
> Mariangela Gualtieri

Besides lighting up and colouring the subjective experience of the world, emotions have also been proven, by neuroscience and psychology, to be one of the main drivers or inhibitors of human action. They play a role in perception and attention mechanisms [53], and there is growing evidence of an inter-dependence of emotional and decision-making processes in the brain [66], especially by observing how damages to parts dealing with the processing of emotional signals lead to an impairment in the choice-making ability [12].

Emotions are also at the core of *affective computing (AC)*, introduced by *Picard* in [78] and defined as *"computing that relates to, arises from, or deliberately influences emotions"*, inspired by the *"essential role of emotion in both human cognition and perception"*. Still in [78], the ability to perceive emotions is also claimed to be a hard requirement for a machine to successfully pass the *Turing Test*.

Slightly drifting away from the realms of theatre and human communication, emotional awareness can bring improvements and enhancements to a large number of existing products and services. One example are for instance recommender systems, which are currently being implemented in many and diverse contexts, ranging from entertainment to education to nutrition; about this topic [99] argues that *"when using applications with recommender systems the user is constantly receiving various stimuli that induce emotive states. These emotions influence, at least partially, the user's decisions on which content to choose. Thus it is important for the recommender system application to detect and make good use of emotive information"*, thus proposing to make the user emotion a *"contextual variable"* in the system.

Another example is instead linked to the use of emotion as regulatory feedback that

makes the agent (being it a software, a robot, a smart device) adapt its behaviour to the perceived user's emotional state, in order not to disrupt it or generate frustration. Possible implementations of this approach can be the *affective reinforcement learning* proposed in [77] for the development of an educational social robot companion or the AI piano teacher described by Picard in the aforementioned paper.

2.1.2. Models of emotion

In the following sections the major computational models of emotion are presented, showing their assumptions and common use cases.

Ekman discrete model

This model belongs to the family of so-called categorical or discrete models, which claim the existence of a *limited number* of basic emotions derived from a number of adaptive responses to prototypical events [90]; it has been introduced by Paul Ekman in [33].

It is based on the assumption that there exist six basic emotions, namely: *anger, disgust, fear, happiness, sadness, surprise*, that are shared by all humans regardless of their cultural background. It also suggests a new meaning for the term *"discrete"*: not only a sharp differentiation of basic emotions but also a discrete differentiation of *universal* emotion patterns and *facial expressions*.

It's also worthy to point out that this initial list of basic emotions has been expanded by Ekman in 1999 [34], with the addition of new emotions not directly linked to facial expressions, such as *amusement, contempt, contentment, embarrassment, excitement, guilt, pride, relief, satisfaction, pleasure, shame*. Subsequent researches also extended the collection, identifying more than 20 "basic" emotions [25, 63].

Circumplex model of affect

This model was introduced by Russell in [86] and is in contrast with the discrete approach described above. Emotions, instead of being each backed by specific and independent neural structures and pathways, are seen as the *"cognitive intepretation of core neural sensations that are the product ot two independent* **neurophysiological** *systems"* [80].

These two systems are mapped into a Cartesian reference system on two orthogonal axis, and represented by the concepts of *arousal* (a sleepiness-alertness continuum) and *valence* (a pleasure-displeasure continuum). Each emotion can therefore be characterised as a linear combination of these dimensions in a continuous way. Figure 2.1 shows an example

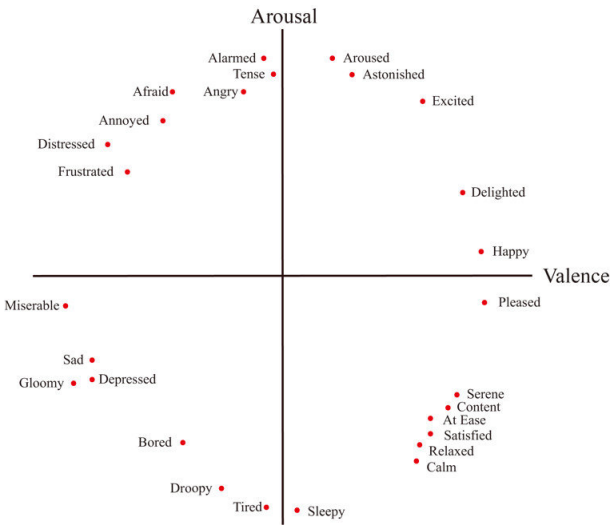


Figure 2.1: Diagram showing how emotions are mapped to valence/arousal pairs in Russel's circumplex model of affect

of how different emotions correspond to different points in the model.

Appraisal theories

Appraisal theories of emotion ditch their neural-backed origin and instead depict emotion as the result of the subjective evaluation (*appraisal*) an individual has of a situation, an object or an event, based on the appraisal of a previous event [91]. The same circumstance can therefore give rise to different emotions in different individuals, according to their beliefs, past experiences and sensory perceptions.

This theory is implemented in [88], which introduces a framework for representing and processing emotional contents in a cognitive architecture. In this context an emotional characterization is added to all cognitive processes and appraisals play a major role.

Plutchik's psychoevolutionary theory

This theory, developed by Robert Plutchik in [79], claims the existence of eight primary emotions, which extend the six of Ekmann's classification with the two new concepts of anticipation and trust. As in fact the name suggests, this theory considers emotions to be linked to survival behaviours developed during evolution (an example is the activation of the fight-or-flight response by fear).

The theory is also based on ten assumptions, which can be summarized in the belief that emotions played a crucial role in helping organisms survive and adapt to the environment, modifying to different forms of expressions in different species; from a set of 8 primary

bipolar emotions all the others can be derived at different levels of intensity or arousal. Figure 2.2 shows the wheel model proposed by Plutchik to describe how different emotions are related, drawing a parallelism with a color wheel to suggest the idea that they can be mixed or exist at different hues and intensities.

Tomkins nine affects theory

This theory has been developed by the psychologist S. Tomkins in [101, 102] as a contribution to the affect theory, and claims the existence of nine primary *affects*, characterised by a pair low/high intensity and physiological expression.

In this context an affect is defined as an innate biological response to different intensities and patterns of neural firings, amplifying the stimulus to direct the attention and motivate action, whereas an emotion is defined as the awareness of an affect combined with the memory of similar situations.

These nine affects are then organized into three distinct categories, on the basis of how they are perceived by the individual, as also shown in figure 2.3:

- *Positive affects*, associated to a perceived reward
- *Negative affects*, associated to a perceived punishment
- *Neutral affects*, not affecting the individual's behaviour

Using this characterization Tomkins also suggested a *Blueprint* for the behaviour of individuals: in order to reach optimal health they are motivated to maximize positive affects and minimize negative ones, and to achieve this it is necessary to properly express them and ease their identification by others.

Selected model

In the end, the Ekman model has been selected to represent the actor's emotions in the context of the framework, both for conceptual and technical reasons. As explained in fact in 1.1.1, the goal of the work is to build a system able to extract relevant features of an ongoing interaction and provide a *preliminary* characterization of its quality and semantics, on which third-party additional modules will build to add semantic layers of increasing complexity, related for instance to the robot's personality and personal appraisal of the situation. For this reason it has been decided to use a simpler (yet expressive enough) model yielding basic emotions, which can be combined and post-processed as needed by other specialized modules.

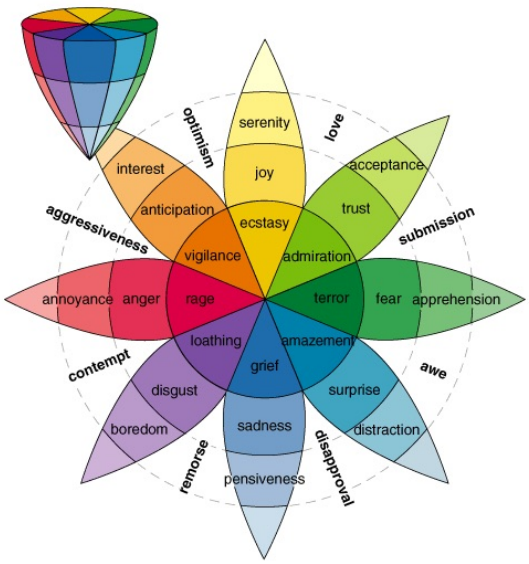


Figure 2.2: Plutchik's wheel of emotions, showing the eight primary emotions and their different intensities, as well as possible dyadic combinations [79]

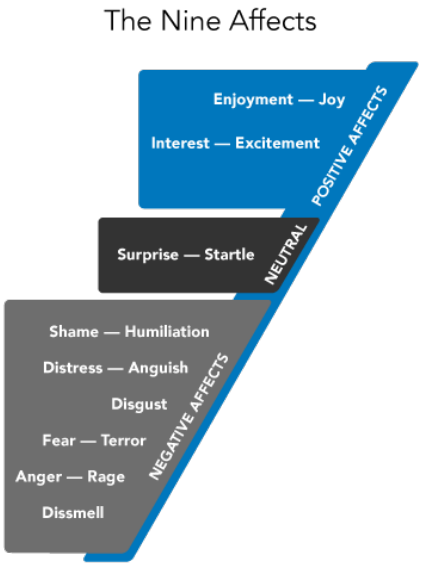


Figure 2.3: Diagram showing the nine primary affects proposed by Tomkins, divided by positive/negative evaluation [40]

The second reason, tied with the implementation of the framework on a physical computational system, is instead related to the larger availability of open-source datasets, algorithms and libraries for emotion recognition based on the Ekman classification rather than other models, and will be covered in detail in chapter 5.

2.2. Body movement

Given the fact, as described in section 1.1, that the developed system focuses on the layers of spatial and visual communication, and in light of the theatrical context, it appears logical to consider the actor's movement as a fundamental expressive medium.

As claimed in [62], *"human movements can be seen as a combination of multiple elements which intrinsically associate meaning, style, and expressiveness"* which reflect not only the intention of the person, but also their state of mind and emotion.

From the analysis of the sources listed in 1.3.1, movement appears as one of the main tools employed to express the intentions of a character, for instance by leaning towards the target of the attention or retreating from something considered harmful or unpleasant; body language and motion cues have been a central element of both cinema and theatre tradition, ranging from the mime in ancient Greece to the *Butoh* in Japan.

Movement is moreover the foundational block of disciplines such as dance, ballet, contact improvisation and choreographic arts in general, whose main characteristic is the use of the body as an expressive medium spanning not only the three geometrical dimensions of space, but also the arrow of time, to organize its movement into a coherent structure onto which the audience can project images, mental states and personal memories.

Differently from other forms of arts employing the body as a narrative support, choreography is more interested in its use as a raw material and instrument, playing with rhythms, levels and constraints to evoke different concepts at a pre-linguistic level. Within this context, several frameworks have been developed to try to organize the large variety of body movements (and sequences of movements) into a single notation system, with the aim of creating the counterpart of a music sheet and provide a reliable and standard mapping of abstract concepts into kinematic utterances.

> *[...] Di fatto si stremava su un colore*
> *o piuttosto sul nome del colore da distendere*
> *sull'omissione, il*
> *mancamento, il vuoto [...]*
>
> Vittorio Sereni - Un posto di vacanza

2.2.1. Laban movement analysis

Among these frameworks is of particular relevance the *Laban Movement Analysis (LMA)*, developed by *R. Laban* [60] and extended by *Ullman, Bartenieff* and *Lamb* to the form that is commonly adopted nowadays.

LMA provides a description of the expressivity of body postures and gestures, to also understand their underlying intention and communicative significance. Moreover, even though LMA primarily focuses on the geometrical properties and motor patterns of human actions, it has been proven able to analyse also their emotional content [82].

Different flavors of LMA exist, focusing on different sets of categories to characterise this expressivity, but the most widely used and taught implementation of Laban theories is the one developed by Bartenieff, who expanded the original two categories (namely *effort* and *spatial harmony*) into a system of four categories usually referred to as *BESS* (from the english initials of the categories, that are explained in the next sections).

Body

Captures a snapshot of the body structure during a movement, describing how different body parts are moving in relation to the body center (usually associated with the navel) and how their motion is mutually influenced

Space

Highlights the connections between body motion and the surrounding space, focusing on spatial patterns, trajectories and spatial tension, considering the traces left by the body in motion as utterances of internal "happenings", such as feelings, emotions and motivations.

Drawing a parallelism with musical scales, movements are organized into harmonious and aesthetically pleasing combinations, inspired also by geometrical considerations on the human body structure, Platonic solids and lines or planes of spatial pulls.

Shape

Describes which shape the body takes during the movement and how it changes, using an euclidean reference frame aligned with an initial position in an egocentric reference system. In this way movements can be characterised along the three cartesian axes and labeled with three sets of bipolar qualities (*i.e. sinking/rising, enclosing/spreading, approaching/retreating*). It is further divided into subcategories:

- *Shape form*: comparing the body shape to a collection of archetypes, like *wall-like, ball-like, pin*
- *Modes of shape change*: describing the relationship between the body and the environment:
  - Shape flow: when the focus of the movement is on the body itself, representing the equivalent of a stream of consciousness leaving the environment in the background
  - Directional: when the body is directed to a certain spot in the environment, with two possible modalities: spoke-like *(i.e. sharp and straight gestures)* and arc-like *(i.e. fluid gestures and bent trajectories)*
  - Carving: when the body is interacting with the whole environment, moving in all the three dimensions during the actions
- *Shape qualities*: using the aforementioned bipolar labels to characterise towards where the body is changing, reflecting at the same time the attitude of the person

towards their surroundings. It is however not concerned about the destination of the movement, but rather on the process with which it is reached.

- *Shape flow support*: related to changes in the internal architecture of the body in the space *(kinesphere)* and to how breath makes it grow or shrink.

> *Movement goes out into space and creates shapes. But also there is inner space, and breath is an inner shaping experience. The body shrinks and grows with each breath.*
>
> Irmgard Bartenieff

Effort

This category qualitatively analyses subtle features of the movements to understand the inner intentions of the performer; it is based on the observation that different intentions have a different influence on the strength, control, and timing of the same movement.

In a similar fashion to the other variables, it is organized into subcategories, called *effort factors*, each having two opposite polarities (*effort elements*), as shown in table 2.1 These factors can be then further grouped at different levels, obtaining different configurations such as *states* (2 factors) and *drives* (3 factors), which are related to different psychological characterizations of the movement. Some examples of drives and states are, in particular:

Effort factor	condensing polarity	indulging polarity
Space	direct	indirect
Weight	strong	light
Time	sudden	sustained
Flow	bound	free

Table 2.1: Table showing LMA effort factors and corresponding polarities

- **Action** drive, combining weight, space and time into typical actions that can be performed to express several emotions
- **Passion** drive, combining flow, weight and time to describe the actor's relationship with space, focus and rationality
- **Awake** and **Dream** states, namely combining space and time and weight and flow. They describe the two opposite polarities of pure thinking and pure emotional feeling qualities of the actor's movement

2.2.2. Computable expressive descriptors of human motion

All the features described in 2.2.1, although useful to give a formal description of human movement and make composers and performers use the same language, are in their original formulation too abstract to be used in a computational model to enable the robot to characterise the expressive content of the actor's movement.

Most of the elements and concepts introduced by LMA are qualitative descriptions of the performer's posture and movements, easily understandable (and executable) by humans but not directly and as straightforwardly computable by a machine.

As in the case of effort, polarities are labeled with terms linked to the common experience of the world a performer could have: a *quick* gesture leaves a feeling and sensation that are perceptually different with respect to a *sustained* movement. A difference, however, that cannot always be broken down into a set of measurable variables and proper thresholds, and therefore not trivial to be detected by machine.

For this reason computational implementations of LMA have been explored, to combine meaningful descriptors into a set of features useful for an automated characterization of the actor's expressions. Apart from the theatre/dance context of the work, an added motivation for the use of an LMA-inspired feature space is to test whether its inherent semantic content about the actor's expressiveness can make an automated classifier outperform a sheer kinetic-only-based movement expression classification.

One of the main works done on computational LMA-implementations is [62], which reviews the descriptors introduced by several other papers, organizing them in levels of abstraction and expressiveness, ranging from basic kinematic formulas such as speed, acceleration and jork to equations that map LMA concepts to the numerical domain.

[17] applied LMA to semantically segment sequences of motion capture data, training a collection of neural networks to classify each motion sequence into three classes *(i.e. indulging, condensing and neutral)* for each effort element. This approach, however, is not directly useful for the goal of this thesis, since LMA characterization is done as black-box by the neural network and not used as an input feature.

[3] focuses instead on recognizing human actions from video images, employing LMA in the feature-extraction step to generate *"compact and informative representations"* to be passed to different classifiers such as a Random Decision Forest, a Multi-Layer Perceptron and a Support Vector Machine. Processing steps are here well explained, and, besides mathematical equations to compute such features from skeletal joints, a pre-processing technique is provided, to convert all the skeletons into the same reference system (centered

in the midpoint of the hips) whose X axis is the segment connecting hips and the Y axis is the spine. Data are acquired using a Microsoft Kinect sensor, yielding a 3D representation of the human skeleton. Proposed formulas can however handle without problems a 2D projection of its points (or skeletons acquired directly in a 2D space).

[104] proposes an LMA-inspired approach to gesture recognition, using LMA-derived local descriptors to generate a soft assignment to a dictionary of reference poses which is eventually used by a Hidden Markov Model for the recognition. As in the previous case, a transformation is applied to the skeletal representation in input, to map it into a common reference system, and a set of equations corresponding to Laban qualities are provided (even though not covering the whole set of descriptors in LMA).

2.3. Proxemics

Lui la lasciò andare, sentì lo spazio freddo che si apriva tra di loro

PETER CAMERON - WHAT HAPPENS AT NIGHT

The movement analysis described so far focuses only on the relationship between the actor's body and the environment, considering for instance how much space is taken up by the body or the dynamic qualities of gestures.

However, since the context of this work is a **paired** improv theatre session, it is important to consider also the role played by *space* in shaping the interaction among the actors and how they in turn use it to convey certain narratives.

Another motivation for these considerations can be found in the analysis of the sources listed in 1.3.1: as many movies, choreographies and even books suggest, spatial layouts and dynamics occurring during an interaction with another person, object or event are usually manifestations of the individual's appraisal and attitude towards them *(e.g. avoiding something fearful, trying to stay close to an object of interest or desire, ...)*.

Considerations that are also psychologically and scientifically backed by the *proxemics*, a subfield of the studies in non-verbal languages concerned with the use and influence of space in human behaviour and communication. This term was introduced by E. Hall in [49], where it is defined as the collection of *"observations and theories about the human use of space"*, which is reflected at different scales in the organization of both micro and macro spaces, from interpersonal daily interactions to the topology of houses and towns.

Proxemics is also deeply linked to culture (Hall depicts it as one of its specialized elaborations), and therefore different people can have different unconsciously coded interpre-

tations of the same interpersonal spatial dynamics; a difference that is often the cause of misunderstanding and failure in intercultural communication [27].

Besides coining the term, Hall introduced in [50] a framework for the classification of interpersonal interactions based on the distance between people, observing how it is influenced by their degree of familiarity [49]. The main idea of this contribution is the existence of discrete social spatial zones surrounding an individual, each characterised by a certain "familiarity threshold" for another person to be let into, as shown in figure 2.4 and summarized in table 2.2.

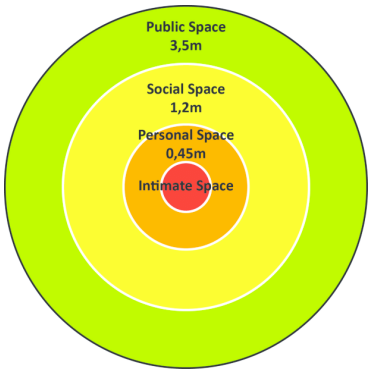


Figure 2.4: Personal zones used in interpersonal human-human interactions [50]

Personal zone	Range	Situation
Close intimate	0 to 0.15m	Lover or close friend touching
Intimate zone	0.15m to 0.45m	Lover or close friend only
Personal zone	0.45 to 1.2m	Conversation between friends
Social zone	1.2m to 3.6m	Conversation to non-friends
Public zone	3.6m +	Public speech making

Table 2.2: Table showing the different personal spaces used by humans in interpersonal interactions and their corresponding paradigmatic situations, according to [61]

Even though the theories introduced so far have been developed to describe and explain interpersonal interactions among humans, they have also been implemented in the context of human-robot interaction.

As described in [31] and [81], many socially-coded behaviours and customs come into play even when interacting with other living or inanimate entities. It therefore makes sense, especially considering the HRI dimension, to implement proxemics awareness into a social robot, to effectively catch the cues that are unconsciously put by the person in the spatial qualities of the interaction to enhance the engagement and mutual understanding.

As it is in fact suggested in [98], this kind of awareness can "*help the design of better models of human-robot interaction, optimizing algorithms for how close robots should approach people*", using as an example Range [56], a digital whiteboard that proactively adapts its behaviour and content on screen according to the proxemics of nearby users. Another example is given in [72], where a reinforcement learning algorithm is proposed to make a robot read "*subconscious body signals*" from the interacting the user (used in particular to characterise their comfort and discomfort) in order to *"adjust interaction distances, gaze meeting, and motion speed and timing"* to "*make the interaction run smoothly*".

2.3.1. Implemented descriptors

Motivated by all the considerations above, proxemics awareness has been added to the framework, computing a high-level description of the spatial dynamics which is further characterised into several specialized sub-qualities.

Preliminary notes on representation

Before explaining in detail which descriptors have been computed, and how, few remarks are needed on spatial data representation, to establish a common ground for the following sections.

Coordinate systems Adhering to the IEEE standard for Robot Map Data Representation and Navigation [46], the robot's position is described in a reference system with the origin in the lower-left corner of the map, as shown in figure 2.5.

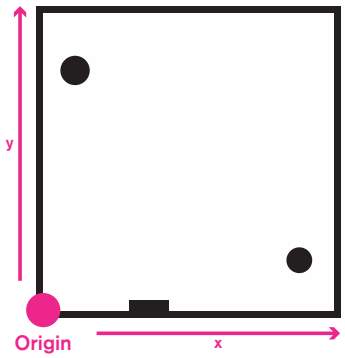


Figure 2.5: Map of the stage used by robot, showing the position of the origin of the reference system used to describe its absolute position

On top of this, the relative position of the actor with respect to the robot is expressed in a reference system centered in the robot, with the x axis exiting from its frontal side and

the y axis from the left side, as shown in figure 2.6.

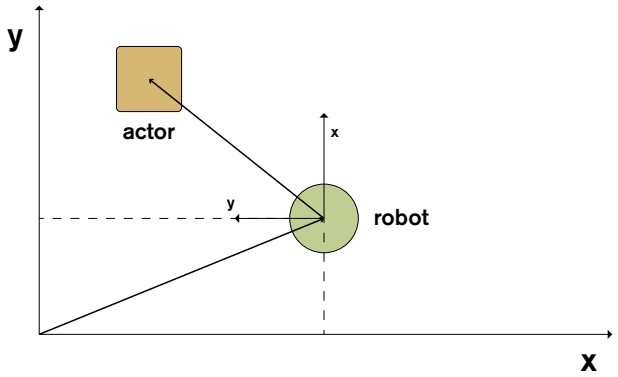


Figure 2.6: Orientation of the robo-centric reference system in which relative position of the actor is expressed

Using this coordinate space, a black-box system is assumed to provide x, y pairs of the detected actor's position (details about its implementation will be given in 5.6.1). These coordinates are then mapped, for ease of computation and to better describe the dynamics of the robot-actor distance segment, to polar coordinates using the equations 2.1 and 2.2.

$$\rho = \sqrt{x^2 + y^2} \tag{2.1}$$

$$\theta = tan^{-1}\left(\frac{y}{x}\right) \tag{2.2}$$

It's worthy to point out that it is possible to switch between these two reference systems them and retrieve the absolute position of the actor in the stage, an information that could come useful to understand and characterise the interaction with the audience and how much of the ongoing action is kept hidden or explicitly shown.

Measurements windowing Data, collected and represented as explained in the paragraph above, are then organized into windows of specific time duration. This approach has been selected because human movements are continuous and usually happen at longer time scales than those at which sensors sample the environment, making **temporal smoothing** a necessary step in the processing pipeline. In particular, collected motion data make sense only if interpreted as sequences for several reasons:

- outliers produced by noise in data acquisition systems, environmental issues *(e.g. bad reflection of laser beams)* or small movements of actor or robot even when still

- different frequencies of sensor sampling and person's movement: human movements are generally happening at larger time scales than those at which sampling devices operate, generating a high number of data points needed to *semantically segment* a single movement, that have to be taken in consideration as a whole to characterise the interaction
- risk of not detecting variations in the actor's position at discrete time *(i.e. compared to the observation at previous time step)* because of their speed being too low with respect to the sampling frequency, generating points separated by a negligible distance.

All these can be condensed in two main points: being able to *correctly capture* and describe complex movement dynamics and *filtering out* unwanted noise and discrepancies.

For the sake of completeness, it should be pointed out that several considerations have been done to understand how to better segment collected motion data, surveying the available state-of-the-art methods.

Several works exist in fact in time series (TS) segmentation (since motion data sequences can be considered a signal evolving through time), based on different techniques such as wavelet transforms [96], convolutional networks applied to image-representation of a TS [51], LMA-descriptors when applied to body gestures [17], or by looking at changes in the signal spectrum.

In testing the available techniques the underlying consideration has been that the activity status of the actor's body is an indicator of their participation in the scene: when the actor is not holding the initiative and is instead waiting for a prompt from the peer, their body is approximately still, without large or expressive movements. For this reason the agitation of body joints has been analysed, either using spectrum- and kinematic-based methods.

Spectrum-based methods attempt to segment a signal by looking at changes in its energy content, an information embedded in the spectrum computed through the Fourier transform. Because of this computational step, however, they are more suitable for signals that are periodic or evolve with a particular frequency. They do not in fact provide useful information for motion data, since it is not a periodic signal and is not characterised by a particular frequency for meaningful periods of time *(i.e. the actor does not usually repeat the same movement back and forth in an oscillating fashion for prolonged periods of time)*.

Experiments on kinematic turn segmentation Another attempt has therefore been to understand whether the analysis of body joints kinematics could instead provide useful

information about the activity of a person. This kind of knowledge would enable the framework to dynamically understand when the initiative is being left by the actor and reply with a prompt to make the interaction develop, a key factor in the success of an improv scene.

For this purpose only the head and the extreme points of limbs *(i.e. wrists and ankles)* have been considered, being the most active body parts when the actor is performing. Using the formulas provided in [62] their velocity and acceleration have been computed for all the frames of a reference video, acquired with a camera mounted on the robot, where a person was interacting with the robot in a *"question and answer"* fashion; single values for each joint have been averaged to obtain a whole-body descriptor at each time step. Plots of these computed kinematic metrics, shown in figure 2.7, clearly reveal the time intervals in which the actor is active and those when is instead waiting for a reply from the robot. In particular, it can be observed that the acceleration provides a cleaner description of the actor's activity.

From a computational perspective this information can be used in an on-line way paired with a simple finite state machine: at each time step the descriptor is computed and if near 0 for a certain number of consecutive steps a "waiting" state is triggered, meaning the robot has to reply to everything the actor has done in between.

To enable this, however, data windowing is still needed, to store all the variables measured from the actor that have to be processed in the robot's turn. Moreover, this method heavily relies on the succesful detection of the person, which cannot be taken for granted, due to both artistic reasons (the actor may decide to perform their action while not seen by the robot) and failures in the actor detection module.

Selected windows length For this reason, to avoid reducing the responsiveness of the system, the final decision has been to process data in overlapping fixed-length windows, whose length has been decided taking into consideration empirical trials and available metrics on the understanding in humans of other people's gestures. In particular, windows have been defined to be long around **1 second**, which is the value offering the best trade-off in terms of system responsiveness and stability of filtered signals and computed descriptors.

Proxemic zones

Inspired by the model shown in figure 2.4, a hierarchy of social boundaries has been added to the model, refining the human-based original formulation to better suit the size and

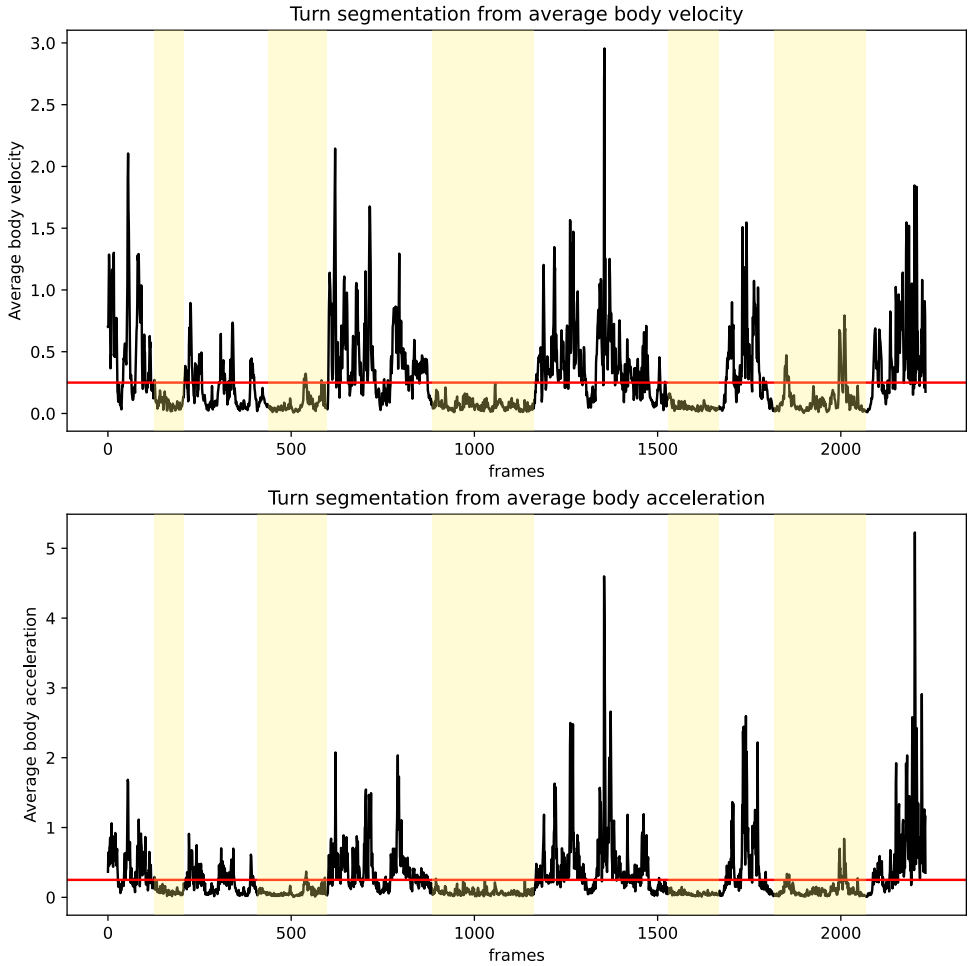


Figure 2.7: Plot of the average body velocity and acceleration computed from head and extreme points of limbs to determine turn segmentations

shape of both the robot and the stage (as described in 1.4).

Values have been in particular refined through several empirical evaluations with people working in AIRLab, who have been asked to interact with the robot while pretending to have different degrees of familiarity, different attitudes and different goals *(e.g. pretending to hold a grudge towards the robot, pretending to be talking to it, pretending not to notice it or to stop an ongoing conversation . . .).*

In the end, 3 zones have been defined, whose corresponding numerical and semantic values are listed in table 2.3

Proxemics zone	Range	Situation
Intimate zone	0m to 0.75m	Deep interaction, sharing of emotions, importance of closeness
Neutral zone	0.75m to 2.5m	Interaction between acquaintances, not characterised by strong emotions
No-interaction zone	2.5m +	No interaction occurs between the actor and the robot

Table 2.3: Table showing the different personal spaces defined for the robot in the developed framework, showing for each the corresponding distance range and example situation

The *intimate zone*, drawing a parallel with the human counterpart, can be considered to be the space reserved to people the robot highly trusts, who are thought to be harmless. At a scenic level, this zone can therefore be used in two different ways: to either depict a positive connection between the actor's and robot's characters, characterised by intimacy, trust and longing, or to show highly negative intentions held by actor's character towards the robot's one, for instance trying to catch or physically hurt it.

The *neutral zone* is instead reserved to all the other situations where the two characters are interacting at moderate levels of energy and intensity. No visible strong emotions or intentions are hence here expressed throughout the interaction and, if present in the mind of one of the two parties, they are kept hidden, for instance in the form of scolding or as a lack of trust and intimacy. For storytelling purposes this zone can therefore be reserved to all the situations in which the characters' relationship has yet to be depicted and explained (not yet heavied by strong events) or for all the "filler" interactions that anticipate a noticeable shift in the unfolding of events.

Lastly, the *no-interaction zone* is an umbrella class for whenever the robot and the actor are not directly interacting. In these cases the two can be seen as independent individuals whose actions and states have a low correlation *(e.g. the emotion expressed by the actor*

could have been caused by an external factor and not by the robot.) However, it is important to point out that this zone can also be used to convey the idea of tension betweeen two characters, where silence and lack of interaction make clear a certain kind of relationship.

Zone classification Using this three discrete zones, the problem of classification can be considered as computing a mapping between the actor's distance and the corresponding social zone. To avoid a zone being selected as a mistake due to noise, an hysteresis-inspired mechanism has been implemented, with a zone being "activated" only if the corresponding function's result remains constant for a certain number of observations.

This smoothing process can be modelled by a finite-state-machine (FSM), composed of as many states as the classified zones *(i.e. 3)* where transitions happen only after a zone is detected for a certain number of consecutive turns, to avoid unwanted fluctuations due to noise in the data. Each state is characterised by its corresponding zone and holds counters for the other zones, which are increased according to incoming observations.

The machine starts in the state corresponding to the *no_ interaction* zone and whenever an observation is collected by the classifier, the proxemic zone computed on the basis of distances defined in table 2.3; this information is then passed to the current state, which yields the new state with three possible outcomes:

- the input zone is the same of the state's one: counters are reset and the current state is kept
- the input zone is different from the state's one: the corresponding counter is increased but is not above the threshold; the current state is kept
- the input zone is different from the state's one: the corresponding counter is increased and is above the threshold; the machine shifts to the state corresponding to the input zone (resetting all internal variables)

Transition thresholds have in particular two different values, to distinguish movements between "near" and "distant" zones: transitions from *intimate* to *neutral* zone requires a lower number of consecutive observations to be activated em (i.e. 3), compared to the transition from *intimate* to *no_ interaction* zone *(i.e. 5 observations)*.

Figure 2.8 shows the architecture of the FSM. For the sake of conciseness, the following notation is adopted: zones are encoded as 0 for *intimate* zone, 1 for *neutral*, 2 for *no_ interaction*; counter for the zone i is represented as c_i; thresholds for near and distant are, namely, K_L and K_H. Arcs are then labeled with a combination of the input zone and conditions on counters.

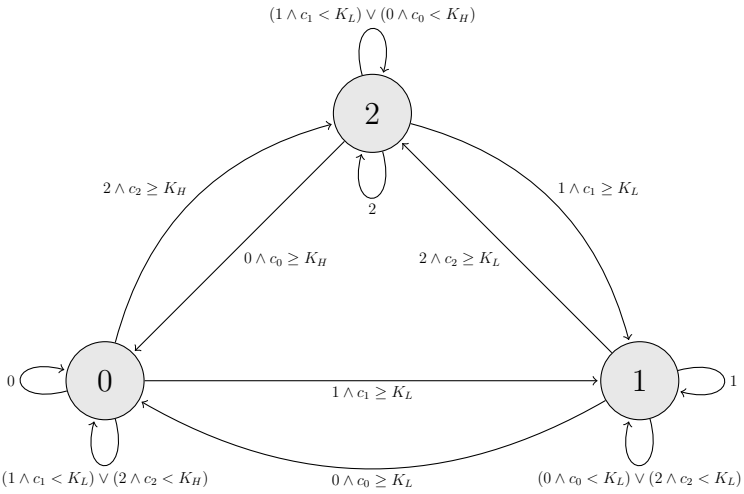


Figure 2.8: Diagram showing the finite state machine designed to compute and smooth the proxemic zone from the measured robot-actor distance

Approach and retreat

In addition to the sheer value of the distance between actor and robot, mapped to a discrete social space, the framework considers also its temporal evolution, to add another semantic layer. Drawing a parallelism with the concept of a derivative, this dimension may enrich the information about the current proxemic zone occupied by the actor, describing the causes of that distance and enabling speculation about present and future attitudes and the levels of fear and trust between the two characters.

As can be seen during everyday interactions, either between humans or even involving animals, the temporal evolution of distance between two individuals is a joint function of the appraisal by both about the ongoing interaction: if one is interested in the interaction (either with positive or negative intentions), the distance will likely reduce; on the other hand if the interaction is evaluated in a negative way or there is still too much uncertainty to draw a conclusion, a higher interspace will be kept.

For all these reasons, also in light of the goal of the framework described in 1.1, this appears to be a reasonable feature for the classification of the scenic actions happening on the stage.

In order to find a proper algorithm to classify whether the person is approaching or retreating from the robot (or if is being "still"), the problem can be formalized as, given a collection of observations of the distance, determining whether their value is generally increasing or decreasing. However, differently from the previous case, where the classification of the zone resembles an hysteresis mechanism, the same FSM-based approach

cannot be used here, since the interest is not in filtering out noise to classify the most recent observation, but rather in filtering out local fluctuations and describe the general evolution inside the considered time window.

Proposed algorithm After several considerations on the more appropriate numerical implementation of this trend classification, the final algorithm has been implemented using a multi-derivative approach. In abstract terms, given a data window composed of M observations, the algorithm estimates $M-1$ discrete derivatives, computing them between the initial point W_0 and data at an increasing time distance (W_i) with the formula in equation 2.3, as shown in figure 2.9.

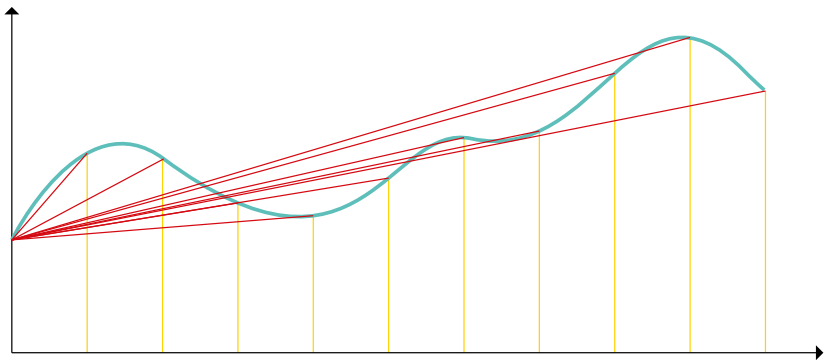


Figure 2.9: Example chart showing the multi-derivative approach employed by the proposed algorithm. A collection of derivatives (red lines) are computed between all the points of the signal and the starting one and the dominant trend is selected through majority voting to subsequently compute the average derivative.

$$\delta_i = \frac{W_i - W_0}{i} \tag{2.3}$$

After computing all the derivatives they are labeled among 3 possible trends, according to the comparison between their value and a reference threshold ϵ. This threshold ϵ is used to define the interval in which the derivative can be approximated to 0 (and hence corresponds to a still trend), and has been empirically tuned to $0.001\frac{m}{s}$ by collecting distance observations of a still subject and estimating the noise in the data.

In particular, trend classes, given a derivative δ can be:

- *Increasing* if $\delta > \epsilon$
- *Still* if $-\epsilon \leq \delta \leq \epsilon$

- *Decreasing* if $\delta < -\epsilon$

Subsequently among these 3 classes the dominant one is extracted through majority voting and returned as the general trend of the window.

Additionally, to estimate the average speed with which the distance follows the detected trend, the algorithm averages the value of all the derivatives belonging to the aforementioned trend and the result is returned.

The code in listing 2.1 shows the Python implementation of the proposed algorithm.

Listing 2.1: Python implementation of the proposed multi-derivative-based algorithm used to classify the trend of distance observations in a given window of data

```
from collections import Counter

def trend_classifier(signal):
    # Compute derivatives
    derivatives = []
    for i in range(1, len(signal)):
        deriv = (signal[i] - signal[0]) / i
        derivatives.append(deriv)
    # Classify trends
    embeddings = [classify_derivative(d) for d in derivatives]
    c = Counter(embeddings)
    # Extract dominant one through majority voting
    trend = c.most_common(1)[0][0]
    # Eventually compute speed
    avg = 0.0
    n = 0
    for i, d in enumerate(derivatives):
        if embeddings[i] == trend:
            avg += d
            n += 1
    speed = avg/n
    return trend, speed
```

Figure 2.10 shows the result of the algorithm applied to distance measurements collected with the robot. Each segment corresponds to a time window and is colored according to the estimated trend: green if increasing, red if decreasing and black if constant.

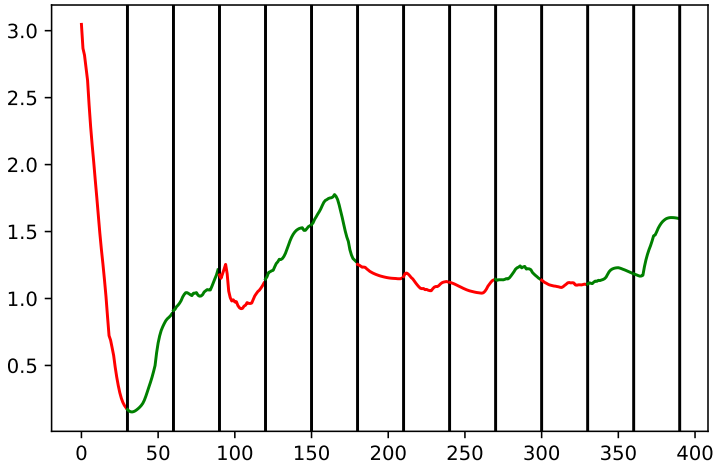


Figure 2.10: Plot showing the classification of distance data (collected with the robot) by the algorithm. Each segment of the curve corresponds to a time window and is colored according to the estimated trend of the signal in that window: green if increasing, red if decreasing and black if constant

Speed

As explained in the previous section, the algorithm designed to characterise the temporal evolution of the actor-robot distance also returns its speed of variation, useful to characterise the urgency of the intentions underlying a certain action or movement.

Drawing a parallelism with the circumplex model of affect introduced in 2.1.2, speed can be compared to the arousal dimension, influencing an action's perceived intensity and in turn the actor's mental state. Speed is in fact a fundamental element in theatre to convey the energy level and motivation of a character; a higher speed, for instance, can be linked to a restless internal state, where the actor's actions are seen as fundamental steps in achieving a certain goal or as something they are highly convinced of.

Employing the algorithm above, speed can be computed along two different directions, as shown in figure 2.11: the segment connecting the robot and the actor, to describe teh urgency and intensity of actions addressed directly and frontally to the robot, and the circumference whose radius is the aforementioned segment, to describe lateral movements and whether the actor wants to stay or escape from the robot's frontal axis.

Speed computed on this circumference can be interpreted as the angular velocity of the actor rotating around the robot, and can hence be obtained by applying the algorithm on the θ angle of the actor's position in the polar representation.

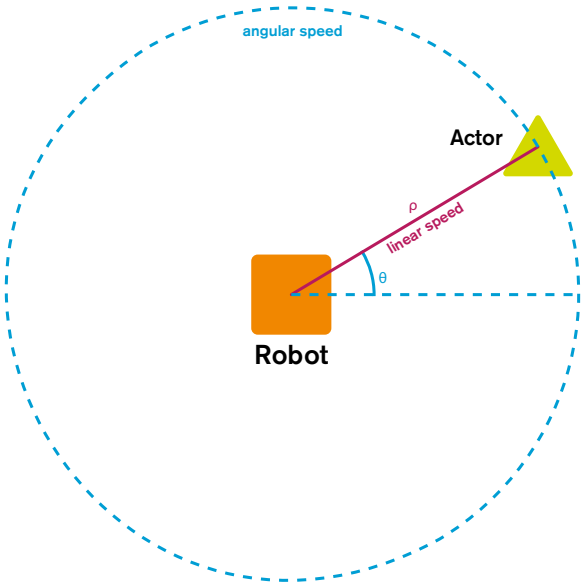


Figure 2.11: Directions considered to compute actor's speed: the segment connecting the robot and the actor and the circumference, centered in the robot, on which the actor lies.

Input data, however, are angular values ranging from $-\pi$ to π (-180 to 180 in degrees), and discontinuities can therefore be present when moving around the robot, as shown by the example plots in figure 2.12. As it can be seen from the right side, a sharp jump is present in the computed angle values when reaching the extreme points of the range, making the trend classification algorithm fail to interpret the dynamics as a single movement in the same direction.

In order to handle these cases, an alignment algorithm has been designed, which is based on the following considerations:

- When considering a movement around the robot, the relevant information is contained in the trend direction, rather than in the sheer values of angles
- This direction can be restored in case of a discontinuity *(i.e. a jump in the plot)* by shifting all the subsequent points by a proper offset
- This offset is constant: depends only on the range in which angular values are represented and on the direction of the jump *(i.e. up or down)*
- The jump direction can be detected from entity of the jump, which can in turn be computed as the difference between the angle at time t and the angle at time $t+1$
- Alignment behaviour *(i.e. amount of offset added to points)* depends on the considered plot region: every jump in the plot causes a *state transition*

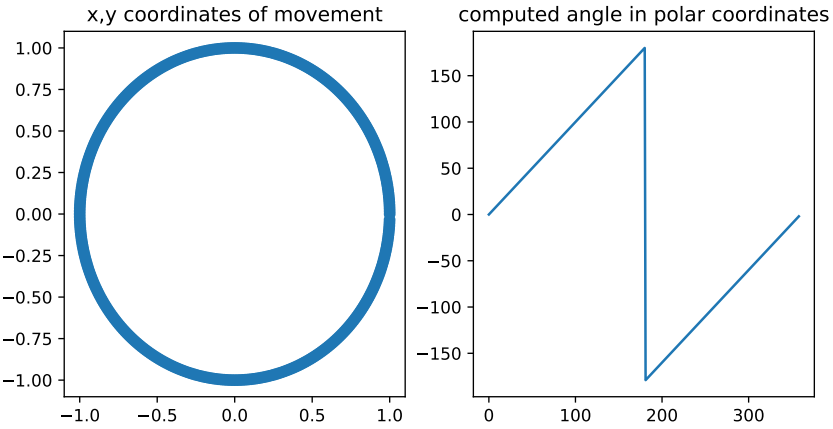


Figure 2.12: Plots showing the discontinuity in the computed angular coordinates when the actor is moving around the robot (a circular movement has been used for the sake of simplicity, without loss of generality). The left side shows collected x, y coordinates, whereas right side shows the θ angle computed using equation 2.2

For this reason, the algorithm has been implemented as a finite-state-machine (FSM), shown in figure 2.13, which in one sweep aligns the data contained in the given time window, applying for each state a different offset to the input value. The FSM is composed of the three following states, whose transitions, as explained before, depend on discontinuities in the data, characterised as the discrete difference δ between two subsequent angle values compared against a fixed threshold to determine the direction:

- *Neutral* (N): the basic case, where data are correct and don't need to be aligned
- *Jump_up* (JU): activated when data suddenly jump from low to high values. In this case data need to be brought back to lower values and hence an offset is subtracted
- *Jump_down* (JD): inverse of the previous state; activated when data suddenly jump from high to low values. In this case, data need to be brought back to higher values and hence an offset is added.

Relative position

In addition to the discrete personal zones around the robot, which can be described as circular crowns, proxemics is also characterised at the level of the relative positioning of the actor around the robot.

By itself, information on the proxemic zone cannot in fact discriminate between points at different angles in the same crown. This knowledge, however, is useful to interpret

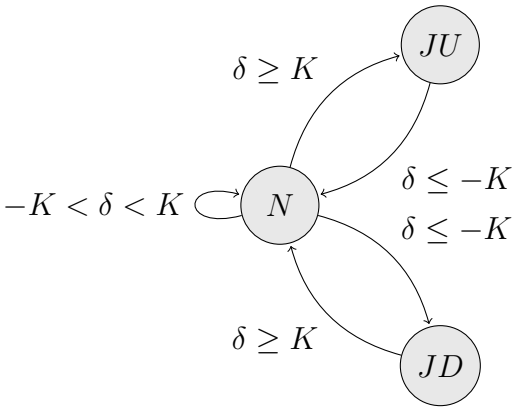


Figure 2.13: Diagram showing the finite state machine designed to align angle observations in presence of discontinuities caused by data representation ranges

different nuances of the same proximity: a very close person can in fact be considered harmful if behind one's back, whereas harmless if in front.

For this reason, four labels have been defined, as shown in figure 2.14, trivially corresponding to the cardinal directions of space. Assignment to one of these areas is then based on the angle θ of the actor's position in polar coordinates.

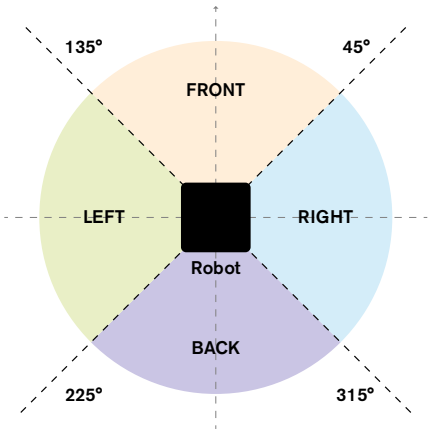


Figure 2.14: Diagram showing the 4 discrete position labels around the robot, along with angles corresponding to their boundaries

Absolute position in the stage

Besides the mutual positioning of the actor and the robot, the theatrical setting adds another dimension to consider. The two performers are not alone on the stage, interacting just with each other, but there is an audience being narrated a story through that same

interaction.

Even if not explicitly addressed, the audience is in fact the vanishing point where, from the stage, all the actions, intentions and dialogues converge; a point from which a higher or lower distance can be kept, to make everything louder or deafer.

For this reason is important for the framework to be aware of the position of both the actor and the robot, to understand at each time if one character wants to hide or reveal a particular action or narrative element, and to also characterise their general attitude *(e.g. distance from the back of the stage is usually considered to be proportional to the extrovertedness or confidence of the character).*

Stage division As explained at the beginning of the section, positions are represented by x, y coordinate pairs in the reference system placed at the lower-left corner of the stage. However, to provide a more immediate description of the relationship of performers with the stage itself and the audience, these two axes have then been discretized into a grid, to be placed as an overlay on the stage and provide discrete labels to salient positions (in a similar fashion to how LMA assign discrete labels to directions of movements).

The used division, in particular, has been inspired by a model of the stage commonly used in acting or dancing, called *Stage Grid*, where the space is tiled in a 3×3 grid, as shown in figure 2.15. Mapping between these two notations can be then easily implemented by just comparing the joint values of x and y coordinates, setting the interval boundaries according to the real dimension of the stage.

Robot position To characterise the robot's position in the stage using the aforementioned model, it is sufficient to know its position in the map's reference system. For this purpose, it is sufficient for the robot to be equipped with a proprioceptive module providing its odometry given the map of the environment.

This odometry is assumed to be computed by a black-box third-party system, since its implementation would go beyond the current scope of this work.

The estimated location on the map can be therefore directly plugged into the discretization function to retrieve the stage position labels.

Actor position In this case, the actor position has to be computed combining the information about the robot's absolute pose and the relative position of the actor in the robocentric reference system described before and shown in figure 2.14.

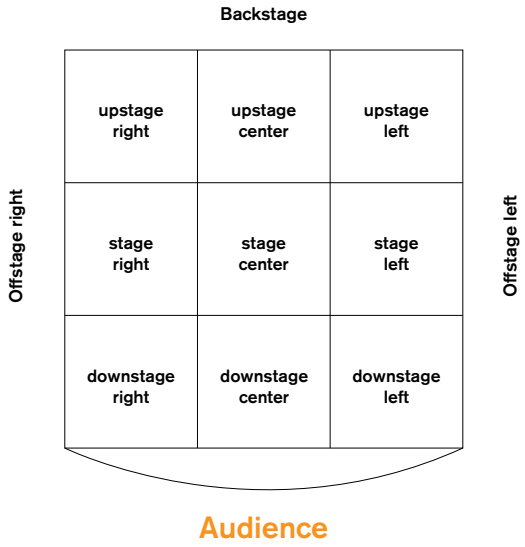


Figure 2.15: Stage Grid model used to divide the stage space into 9 discrete tiles, showing the label associated to each tile

In particular, knowing these two information, a simple coordinate change, described by equation 2.4, yields the actor's position in the map's reference system, from which to derive the discrete stage label. In the equation, $\vec{r_S}$ and $\vec{a_S}$ represent the position vectors of namely the robot and the actor in the stage coordinate system, while $\vec{a_R}$ is the position vector of the actor in the robo-centric reference system. A minus sign is in particular added to $\vec{a_R}$ to take into account the opposite orientation of the x axis in the robot coordinate system with respect to the stage's one.

$$\begin{aligned} \vec{r_S} &= r_{S,x}\hat{x} + r_{S,y}\hat{y} \\ \vec{a_R} &= a_{R,x}\hat{x} + a_{R,y}\hat{y} \\ \vec{a_S} &= (r_{S,x} - a_{R,x})\hat{x} + (r_{S,y} + a_{R,y})\hat{y} \end{aligned} \tag{2.4}$$

2.4. Eye contact

I saw happiness and pain in your eyes
and reflection of the Paradises lost
and regained and lost again

Jonas Mekas - As I Was Moving Ahead Occasionally I Saw Brief Glimpses of Beauty

The last feature that has been identified as relevant for the framework is the "eye contact" between the robot and the actor (intended in this case as the two performers facing each other, with the actor's gaze directed at the robot).

Eye contact, or *gazing*, is in fact, together with proxemics, posture and facial expressions, one of the fundamental cues of non-verbal communication. Its presence or absence conveys a variety of feelings, and helps to characterise different situations, understanding for instance whether an action (or an utterance in general) is directed to the other character or to another point in the stage.

Analyzing the sources listed in 1.3.1 it has been observed that eye contact is often employed in storytelling to give information about the following elements:

- where the attention of a character is directed, adding information to the target of their action or inner drives
- whether a character is trying to hide some actions *(i.e. doing something while not in the view field of another character)* or on the contrary wants them to be explicitly seen *(i.e. doing something only when inside the view field)*
- presence or absence of an ongoing communication between two characters
- positivity of feelings and level of trust and intimacy between two characters (mutual connection or resentment/avoidance)

For these reasons possible methods to detect eye contact have been explored, which will be described in detail in chapter 5, as well as their corresponding computational representation of the eye contact presence.

3 | Output space

Having all the input features been described, this chapter considers how they are jointly processed to produce the framework's output.

3.1. Methodology

As explained in 1, the framework is expected to produce a classification of the ongoing scenic actions in the context of a paired human-robot theatre improv scene.

To achieve this, the work has been organized as follows: understanding in the first place what the concept of scenic actions and how their saliency is characterised; defining in detail the output space, listing the relevant scenic actions that should be detected by the framework (or their general archetypes); deciding a proper representation and encoding of each action; surveying and eventually choosing how features are combined to produce a resulting action and which classification method to employ among the available ones, showing advantages and disadvantages for each.

It should be pointed out that implementation details concerning this very last step will be described in detail in chapter 5, to follow the same structure adopted for the description of the feature space.

3.2. What's in a scenic action?

In the context of this work, a *scenic action* has been defined as a combination of input features (hence observed from an actor on stage) that carries a specific and self-contained meaning that timely characterises the context of an ongoing improv interaction and the actor's internal feelings and emotions.

This concept of self-containment is related to the goal of the system explained in 1.1 and the hierarchy of semantic and semiotic layers outlined in 1.1.1. To achieve its goal, the framework should in fact be able to characterise current interactions solely on the basis of the perceived dimensions described in 2, using their temporal evolution as the criterion

to discern different moments and contexts of the scene.

Production of scenic actions is conceptually placed one step before contextual awareness. Drawing a parallelism with neuroscience, this characterization can be considered similar to neural activations produced by the amygdala, which has been shown to play *a critical role in linking external stimuli to defense responses* [64] and is responsible of the *fight-or-flight* mechanism, which comes into play before the activation of rational circuits.

In light of these considerations, scenic actions represent a partial perceptual response, framing observed cues and stimuli inside a vocabulary of archetypical interactions, which can be therefore thought of as output classes of a multi-class classifier.

3.3. Relevant scenic actions

Starting from the definition given in the previous section, scenic actions identified by the framework have been defined through an approach inspired by the circumplex model of affect (explained in 2.1.2).

At a conceptual level, scenic actions can in fact be seen as labels appointed to hyper regions in the multi-dimensional input space of the framework. Because of this multi-dimensionality, it is impossible to have a 2D diagram similar to the one proposed by Russel (shown in figure 2.1), and the labelling has therefore been organized in a cascade fashion, that resembles that of decision trees, splitting at each level of all the possible values of the input features as shown in figure 3.1.

Besides this top-down direction, giving a name to different combinations of input features, the search has been also conducted in a bottom-up fashion, surveying relevant and paradigmatic scenic actions both in everyday life and in a theatrical context, which have been then decomposed into the corresponding input values.

In the end, the classification reported in table 3.1 has been developed, using as relevant features the perceived emotion, the proxemic zone occupied by the actor and their movement in relation to the robot (speed and direction). In particular, for the sake of simplicity, instead of splitting features on the basis of their numerical values, linguistic labels have been used, which can then be mapped to the proper numerical ranges.

It's important to point out that not all the features listed in chapter 2 have been used as discriminant attributes. This has been done to keep a reasonable small size of the output space, also in light of the considerations made above, and the biological parallelism with instinctive brain reactions, avoiding a too rich characterization of actions and separating

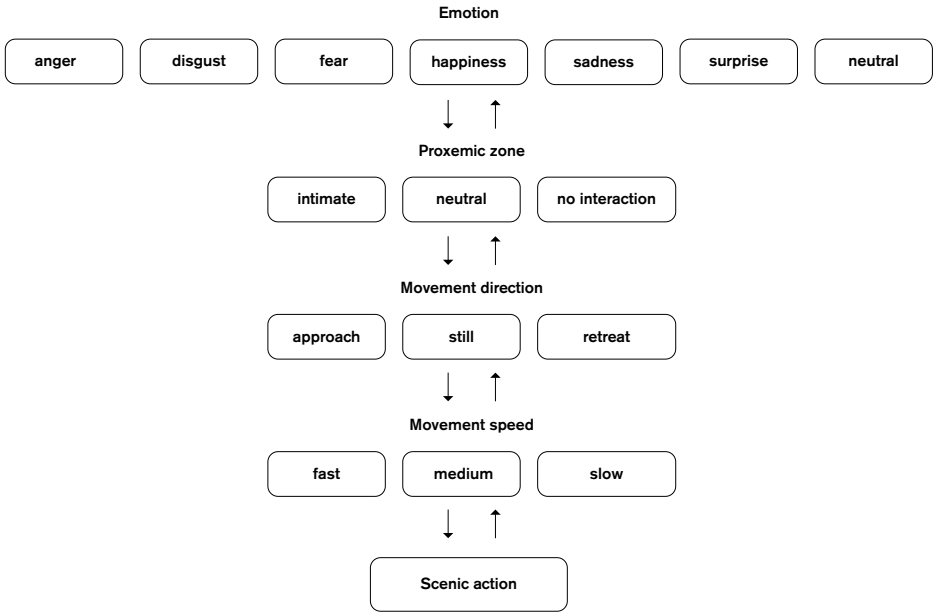


Figure 3.1: Diagram showing the approach used to define scenic actionson the basis of available features, highlighting the top-down and bottom-up directions

instead the two different levels of description (leaving to other third-party systems more linked to the character's personality the task of adding nuance to each action). Features not directly used in the classification, however, are still made available as outputs of the framework, to be used by other systems for more complex elaborations.

3.4. Action representation

Scenic actions, besides being the final output of the framework, also represent the communication interface with other systems that can be deployed on the robot, controlling its personality and choosing the proper replies to each action.

Another point in the work has therefore been to define their computational representation, namely the data structure used to represent and exchange them.

Because of the limited available options, the final choice has been to represent them using a shared vocabulary, encoding each action with a numerical index (*i.e. an unsigned integer*). In this way, it will be possible to extend the list in the future in a transparent way for all the consumer systems: employing a factory design pattern each of them can in fact dynamically instantiate the proper computational representation of an action, supporting the full vocabulary or a reduced set (for instance merging multiple actions under the same

Emotion	Direction	Speed	Zone	Scenic Action
Anger	approach	fast	intimate	Attack
	approach	slow / med	intimate	Intimidation
	still	-	neutral	Scolding
	retreat	any	no int	Holding grudge
Disgust	retreat / still	any	neutral / no int	Refuse
	approach	any	any	perplexity
Fear	approach / still	fast / medium	intimate / neutral	Share fear
	approach	slow	intimate / neutral	Caution
	still	-	neutral / no int	Hesitancy
	retreat	slow	any	Shock
	retreat	medium / fast	any	Escape
Happiness	-	-	intimate	Share joy
	-	-	neutral	Greet
	-	-	no int	Happy person
	retreat	any	neutral / no int	Satisfaction
Sadness	approach	any	intimate	Share sadness
	-	-	neutral / no int	Sad person
	retreat	any	neutral / no int	Disappointment
Surprise	approach / still	any	intimate / neutral	Share surprise
	still	-	no int	Astonishment
	retreat	any	neutral / no int	Disbelief
None	any	any	any	Neutral

Table 3.1: Table listing all the relevant scenic actions identified to be outputs of the framework. The first four columns represent values of input features, while the last column indicates the identified action. Features' values are expressed as high-level discrete labels which will be later numerically groundeds

representation or using only those relevant to its specific goal).

3.5. Available classification methods

Eventually, possible methods to carry out the described classification have been explored. In particular, one can observe that in the end the problem of determining a certain scenic action can be reduced to a multi-class classification problem on the basis of the input features listed above in 3.3. For this reason, all the available classification methods can theoretically be employed for the task, but each has its own pros and cons that have to be taken into account.

The following sections will consider a selection of those methods, analyzing the tradeoffs of each to eventually explain the final decision.

3.5.1. Neural networks

The first possibility is the use of a neural network to classify the scenic action from the set of observations about the actor and the stage.

This solution has, on one hand, the advantage that no manual definition of feature/action mapping would be required, leaving to the implemented model the burden of understanding which features (and combinations of features) are relevant for the activation of a specific action. On the other hand, however, a major drawback is that the model requires a proper dataset to be trained on. This dataset would have to be created from scratch, acquiring samples of all the classifiable actions from different subjects to ensure a good variability and representativeness of expressions.

For this reason implementation of such a model would require additional work that would go beyond the scope and timings of this thesis, and has therefore been discarded.

3.5.2. Decision trees

A second solution is the use of decision trees and random forests, already adopted in several works in the literature for gesture or emotion recognition [3, 37, 110].

These models present the advantage that they can be built even in the absence of a training dataset, manually defining the split point and adjusting according to empirical tests and observations, therefore removing the need of creating a new dataset.

Because of their mechanism, however, they discretize the feature space into sharp regions, each associated with a class label, and a small variation in the input features may cause an abrupt change in the final output. For this reason, it would be necessary to add a filtering and smoothing mechanism at the end of the action classification pipeline, to ensure an acceptable smoothness and stability of the predictions.

3.5.3. (Fuzzy) rule based classification system

To overcome this limitation and make the system react smoothly to changes in the input features, the third (and eventually chosen) option is the use of a fuzzy classifier whose functioning and theoretical background will be explained in detail in section 5.7.

The use of a fuzzy rule system is in particular a common technique in the domain of mobile interactive robotics and, as stated in [13], *"probabilistic fuzzy rules can handle inconsistent behavioural data pattern"*, providing a smoother and stabler output of the system.

4 | System deployment

Before proceeding to show how all the feature extraction, processing and classification steps explained in the previous parts of the document have been implemented, this chapter presents the reference robot on which the framework has been deployed and how the system architecture has been designed, to clarify details, terms and concepts that will be used in the rest of the document.

4.1. Reference robot

Instead of designing and building a new one from scratch, the reference robot on which to deploy and test the framework has been chosen among those already present in AIRLab. Choice, in particular, fell on Robocchio, developed in [15] and shown in figure 4.1, already targeting the theatrical domain and therefore built by taking into account expressive capabilities and features. As stated in the introduction, the robot *"is capable of expressing emotions through the movement of the eyes and pelvis and moving along the stage. [...] The robot is programmed to follow a script containing all the lines and the actions that it must perform to play its role and express its emotions."*

4.1.1. Robot's platform and body

Robocchio's body is based on a pre-existing platform named *TriskarOne*, developed in AIRLab for the RoboTower project [14], an interactive human-robot game in which users have to take control of plastic towers while the robot tries to anticipate players' movement and protect the towers.

The structure of the platform is triangular, with an omniwheel on each vertex and all the needed components (e.g. power units, motor drivers, laser scanners, electronics) placed inside and surrounded by plastic bumpers for protection. A vertical shaft then springs from the base center, representing the robot's spine and hosting the computational units and the speakers.

The upper part of the body is inspired by the kinematic of the Blossom robot [97], based

Figure 4.1: External appearance of the Robocchio robot [15] used as the reference deployment platform of the developed system

on a structure elastically suspended to a central pole, whose tilting is controlled through wires and pulleys. This approach enables several expressive possibilities, such as leaning forward or backward but also brings about possible issues in terms of stability, especially for the camera, due to the wobbling of the elastic structure after a movement.

A peculiarity of this robot is the eyes, shown in figure 4.3. They are composed of a polystyrene sphere (the eyeball) connected to a PVC elastic pipe. This kind of structure allows the eyes to rotate and look in different directions (the field of movement is declared approximable to a spherical cap), adding nuance to the robot's expression.

4.2. Existing hardware components

To achieve its goal Robocchio's platform offers several sensors and actuators, such as a microphone, used to detect pauses in the actor's speech and trigger a reply; a speaker, to output the voice; motor drivers, to move the wheels, and two laser sensors, to map the environment and localize the robot in it.

In the following paragraphs, a brief description of the relevant features of each component is given, to highlight in which way they are useful for the implementation, and to list unnecessary ones.

Figure 4.2: Snapshot of a session of the RoboTower game [14]

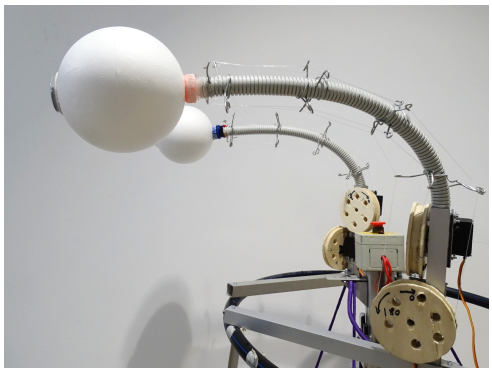


Figure 4.3: Eye structure and moving mechanism mounted on the reference robot

4.2.1. Audio devices

The robot is able to emit sound thanks to two 2.5W amplified speakers with aux input connected via USB to the power supply, whereas listening capabilities are provided by an omnidirectional condenser microphone with a range of about 5 meters. Both components are mounted on the central shaft of the robot's body and connected to the main computer through an external USB sound card to reduce both input and output noise.

However, as explained in 1.1.1, the verbal communication layer is outside the scope of the version of the framework currently being described, hence audio capabilities are not strictly required by the implemented version of the system.

4.2.2. Laser scanners

In order to perform localization and mapping in the stage, the robot mounts 2 *Hokuyo URG-04LX-UG01* laser scanners, shown in figure 4.4. Each sensor, intended for indoor use, is able to scan a semicircle of 240°, with a maximum radius of 4 meters, outputting the measured distance for each point at steps of 0.36°, as shown in figure 4.5.

Figure 4.4: Hokuyo URG-04LX-UG01 laser sensor mounted on the reference robot

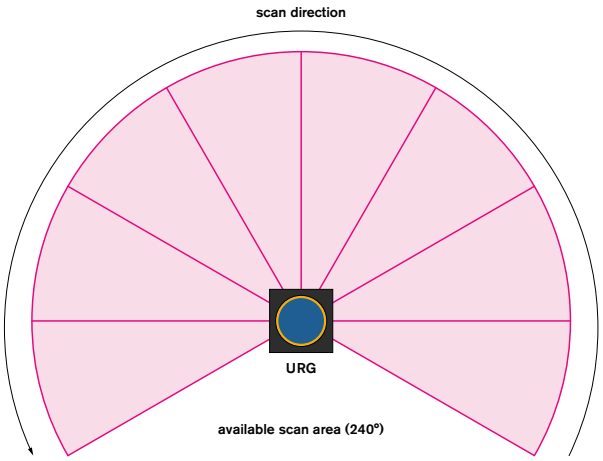


Figure 4.5: Diagram showing the scanning area of a single mounted laser sensor

Distance is in particular measured by computing the phase difference between emitted and received signals (infrared laser beams), and for this reason, as reported in the official documentation, *"it is possible that completely causal and unexpected measurements may occur due to dust or the passage of small objects"*.

Lasers commonly suffer from transparent or dark surfaces, that cause the emitted beam not to come back to the source (and hence making the robot think no obstacle is present in that direction), or from irregularities in walls that cause a chain reaction of reflections, leading to the same issue. However, it is worth to underline that the setup of the stage described in 1.4 reduces the probability of these issues since it does not feature any irregularity in the boundaries or reflectant elements.

As explained at the beginning, the robot mounts a *pair* of sensors. This is because a single sensor would not be sufficient to cover the whole area surrounding the robot, due to the presence of blind spots. This configuration, however, causes an overlap of the scanning

areas, and the two incoming observations need to be merged into a single 360° wide set of measurements.

For this purpose, a specific third-party software component is integrated, which, given in input the two data streams of the laser sensors produces a single joint stream of laser scans. More specific implementation details will be given in 4.5, where existing software components are analysed.

4.2.3. Motors

The robot mounts different types of motors, used to control different body parts.

Eyes and body trunk movements are obtained by servo motors because they are easier to move at a specific angle and offer a better cost/effectiveness ratio when compared to traditional DC motors.

The robot's base, instead, because of its triangular shape described before, requires at least three DC motors and omniwheels *(able to move in any direction)*, implementing a holonomic kinematics, with the configuration shown in 4.6.

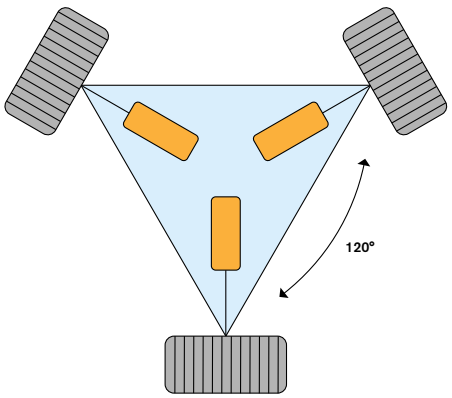


Figure 4.6: Diagram showing the configuration of motors and omniwheels in the robot's base, where each motor is placed every $\frac{2\pi}{3}$ on a circumference and wheels are orthogonal to motors' axes

In particular, Robocchio mounts three MAXON 118798 DC motors, each equipped with an encoder to track the current rotation of the wheel by means of a *hall-effect* encoder. Motors are powered at a voltage of 24V and power of 70W, allowing the robot to reach a maximum speed of 1.4m/sec.

4.2.4. Computational units

The system implemented on Robocchio makes use of two different computational units following a master-slave pattern, to detach the control of servo motors and battery from the main software.

Namely, the implemented software and control system runs on a Shuttle DH310 Mini PC, shown in figure 4.7, featuring an Intel I7-8700 CPU, 8GB DDR4 RAM and 240GB, whereas the control of servo-motors and battery status is delegated to an Arduino Uno board, shown in figure 4.8, based on the ATmega328P microcontroller.

Figure 4.7: Shuttle DH310 Mini PC powering the Robocchio robot

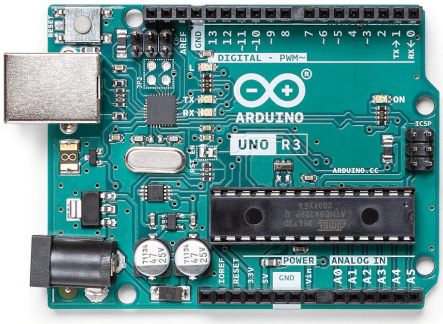


Figure 4.8: Arduino Uno board controlling servo motors and battery status

DC motors are managed through Nova Core modules, shown in figure 4.9, driver boards based on the STM32 chip on which a PID is implemented, modeling the motors as second-order systems having as input the applied voltage and as output the angular speed.

4.2.5. Power supply

Given its mobility and the need to move freely on the stage, the robot's components are powered by 2 lead batteries, each with a nominal voltage of 12V and a capacity of 9Ah. Their charge level is periodically monitored by the Arduino board through a voltage divider circuit, emitting a buzzer sound when it falls below a certain threshold.

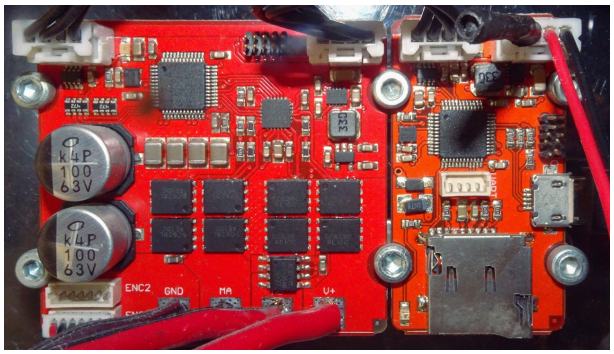


Figure 4.9: Nova Core driver controlling DC motors

For development and debugging purposes, where the robot doesn't need to move but has instead to stay still and turned on for prolonged periods of time, batteries can be replaced by a direct connection to the electrical grid.

4.3. Added hardware components

All the hardware components described in the previous section, even if not strictly needed by the framework to realize all its sensing capabilities, have been retained in the robot, to be used by the third-party actuator systems to physically express and convey an appropriate response to the detected actor's scenic actions.

The existing set of sensors, however, was not sufficient to provide all the needed perceptions and observations, especially those based on visual cues, and hence an external camera has been added as a new hardware component.

The camera is in particular a Logitech C310 HD USB webcam, shown in figure 4.10, with a resolution of 720p, a frame-rate of 30 FPS and a sensor of 1.2 mega-pixel. It mounts a fixed-focus plastic lens with a diagonal field of view of 60 degrees, which has been extended by using an external wide-angle lens.

Figure 4.10: Logitech C310 HD webcam added as a camera to the reference robot

4.4. ROS

Before moving to consider all the existing software components controlling the behaviour of Robocchio, this section analyses the framework they are implemented with.

One of the main requirements of the original software was in fact to be able to control all the robot movements in parallel and to implement as many of its features through pre-existing libraries; for these reasons, the ROS framework was selected.

As it can be read in [15], motivations for this choice range from the popularity of ROS in the field of robotics to the large availability of third-party libraries, to the simple and transparent support of multitasking and parallel computing to the support of both Python and C++ programming languages.

ROS is the acronym for *Robot Operating System* and, as the name suggests, it aims to be a full-fledged meta-operating system for a generic robot, providing hardware abstraction, low-level device control, message-passing logic and pre-implemented commonly-used functions. Moreover, as can be read in the official documentation, one of its main goals is not to be the most complete robot framework, offering the widest set of features, but instead to let developers reach the largest amount of code reuse across the implemented system, building on the concepts explained below.

It's important to note here that even though version 2 of the ROS framework has been released, Robocchio is implemented using ROS1 and hence the following considerations are referred only to this specific version. Description and analysis of the ROS2 architecture are given in 4.4.2.

4.4.1. Basic concepts

ROS is built upon three levels of concepts, which describe how the framework's files and artefacts are organized on the machine, how the community is handled and how all the features and implemented systems are modelled and managed. For the scope of the work it is relevant to briefly introduce only some concepts in the *Computation Graph* level, which represents all the processes interacting to produce, transfer and handle data:

- **Nodes** are the foundational blocks to build every system in ROS, and they explain the term *graph* in the *computation graph*. They are processes performing computation, designed using a principle of fine-grained modularity. For this reason, a robot control system is usually composed of a large number of highly specialized nodes *(e.g. one for localization, one for planning . . .)*

- **Master** is a higher-level node providing name registration and lookup to all the other components in the computation graph, in a centralized fashion
- **Messages** are the data structures through which nodes communicate and exchange data and results. They can either be built-in primitives or custom-defined objects, supporting arbitrary nesting of fields.
- **Topics** are the communication channels on which messages are exchanged. Each channel is characterised by a name and the type of transported message (a topic can only transport messages of a single type) and is based on a publish/subscribe mechanism, with each node being able to publish and subscribe to an arbitrary number of topics. This model has been chosen because it allows to develop each system in a transparent way, decoupling the production of information from its consumption thus making nodes ideally interact as black-box systems.
- **Services** are a concept similar to a remote procedure call, and have been designed to implement a request/reply paradigm to make nodes fulfil operations in the distributed system without relying on the topics' paradigm. They are defined by a pair of message structures: the request and the reply, and each node can use the service by sending the request message and waiting for the corresponding reply.

Before moving to consider how these concepts have been used in the implementation of Robocchio's control system, some remarks have to be made about the current state of the ROS framework.

4.4.2. ROS2

Since the initial release of ROS in 2007, the community started working in 2014 on a refactoring, to keep up with changing needs in the robotics community and to improve the overall quality of the framework. Motivations are in particular multi-faceted: enabling new use cases such as multi-agent architectures, deployment on embedded platforms, real-time systems and networking fault tolerance; replacing custom implemented parts of the architecture (such as the communication layer) with new third-party technologies that became the de-facto standard in several fields; improving the design of the API interface.

As it can be read in the design documents of ROS2, achieving this goal while extending the existing ROS codebase would have required too many intrusive and critical changes, with the risk of rendering the product unreliable for all the existing users. For this reason, a parallel set of packages has been developed, under the name of ROS2, which can be installed alongside and interoperate with ROS1.

Interoperability, in particular, is guaranteed by the ROS1 bridge project [1], a package that enables the exchange of messages between ROS1 and ROS2 nodes by mapping messages published on a certain topic into the respective data structures. The bridge can work out-of-the-box with pre-built primitive interfaces defined by the ROS framework, but can also support the conversion of custom messages by building it from source after having built and sourced the needed custom message types in the workspaces.

At a conceptual level, the architecture of a ROS2 system is quite similar to its implementation with ROS1, having the concepts of nodes, topics and messages being retained. Major changes are in fact at a lower level, related to the networking architecture, lifecycle management and build system, which can be considered transparent when developing each component; moreover, porting of simple packages to ROS2 (as in the case of the Robocchio control system) can be achieved in a relatively short time with few code changes (mostly related to syntax changes and different API interfaces).

For all these reasons the new system described in this thesis has been implemented using ROS2, employing the aforementioned bridge to make it communicate with legacy parts of the architecture, whose porting would have been outside the scope of the work or that make use of libraries and functionalities not yet ported to the new version of ROS.

4.5. Existing software components

Before designing and implementing the new sensing system on the reference robot the currently deployed software has been analysed, to understand which parts could be useful for all the purposes described so far and hence be retained.

The goal of the previous system was to make the robot able to interpret a fixed script, loaded in properly-encoded file on the machine. Several components are therefore related to script and turn management, as well as the control of physical moving parts of the robot's body, and are not a fundamental part of this work (being the actuation of physical replies to the estimated action delegated to a third-party system).

What is interesting to retain are instead the perceptive and proprioceptive components, providing information about the surrounding environment and the robot's relation to it. They are in particular the odometry and laser sensing components, which are described in the following paragraphs.

4.5.1. Odometry

Odometry is the estimation of the robot's movement and change in position relative to a fixed reference position, computed from kinematic formulas governing its motion. Knowledge of the robot's position in the stage is relevant in the context of the work for two main reasons:

- understanding and characterizing the relationship with the stage and the audience of both the robot and the actor, in the context of the stage grid division explained in 2.3.1
- informing the actuator system and preventing it from performing unsafe movements that could make the robot go out or even fall from the stage

Odometry computation is implemented in a standard way using the built-in ROS data types by the *odometry_publisher* node, which computes the robot's pose from its planar and angular velocities, as represented at a high level in equation 4.1.

$$\langle X, Y, \theta \rangle = Odom\,(V_x, V_y, V_\omega) \tag{4.1}$$

At each time step (represented as the time interval δt), the movement is computed from current velocities and the estimated pose is updated using the formulas in equation 4.2

$$\begin{aligned} \delta\theta &= \omega \cdot \delta t \\ \delta x &= (V_x \cdot cos(\theta) - V_y \cdot sin(\theta)) \cdot \delta t \\ \delta y &= (V_x \cdot sin(\theta) + V_y \cdot cos(\theta)) \cdot \delta t \\ \theta &= \theta + \delta\theta \\ X &= X + \delta x \\ Y &= Y + \delta y \end{aligned} \tag{4.2}$$

4.5.2. Laser sensing and merging

Acquisition of laser scans from sensors is realized through the *urg_node* ROS1 package, which reads data from a serial connection and publishes them on two topics, namely for the left and right sensor. As in fact already explained in 4.2.2, two lasers are present

on board of the robot, making it necessary to computationally merge the two streams of scans.

This problem is solved by the *ira_laser_tools* package [10], developed in the IRALab laboratory of MilanoBicocca university which, as stated by the official website, *allows to easily and dynamically (rqt_reconfigure) merge multiple, same time, single scanning plane, laser scans into a single one.*

From an implementation perspective the package subscribes to the two incoming laser scan topics *scanLeft, scanRight* (namely corresponding to left and right sensors) and publishes merged observations into a third topic *scan*, in a way completely transparent to the remainder of the system, which can just treat this data as the real ones, as if the robot had only one 360°-wide laser sensor placed halfway between the two.

4.5.3. Localization

Using odometry and laser scans, the robot is able to localize inside a pre-loaded map of the environment (built using the *gmapping* ROS library). Localization is in particular achieved with the AMCL ROS package, a probabilistic localization system for a planar robot based on an adaptive Monte Carlo localization approach, using a particle filter to track the pose of the robot on the given map.

The internal parameters of AMCL have already been tuned during the development of the previous system. Since no modification has been done to the robot's kinematics and structure, no fine-tuning was necessary and parameters have been left untouched.

4.6. Notes on the designed architecture

To understand which new components were necessary to be designed and developed, features identified and described in chapter 2 have been used as requirements and functionalities. Following the principles of modularity and separation of concerns, each feature has been thought of as captured or processed by a specific ROS node.

Since ROS2 has been chosen for the implementation, new components had to be developed in a separate architecture, not as an extension of the original Robocchio control system (though the two systems can still interact thanks to the modular approach of ROS). For the sake of clarity, specific details about the final architecture's topology will be given at the end of the next chapter, after having described how each functionality has been implemented, to clearly show the nodes' mutual dependencies, data flows and design decisions that need a prior explanation of context- and implementation-related motivations.

5 | System design and implementation

This chapter explains in detail which models, algorithms and technologies have been selected and implemented to extract and process the features described in chapter 2, showing also how they are deployed as ROS nodes in the final architecture, which is described at the end of the chapter.

5.1. Preliminary notes on data collection

Before describing how each module of the architecture has been actually implemented, this section explains some common design choices and concepts that are shared across the whole system.

5.1.1. Data windowing

The robot's system senses and observes the environment and the human actor in a continuous way to build its classification, collecting a large amount of data for each second of the interaction.

For this reason, updating the classification on the basis of just every single incoming input may lead to unstable and incoherent behaviour. Moreover, the semantics of human expressions is often perceived and understood at frequencies lower than those of sensors' data acquisition, and single data points are hence not able to completely characterise (and also help disambiguate) a single expression, especially if treated as independent observations without considering the recent history; expressions can then be decomposed into simpler movements, that could not carry a very poignant meaning by themselves, but make sense together only when considering their temporal dynamics.

One example to clarify could be the case where the actor's aim is to keep the maximum distance possible from the robot: in this situation there could be a moment in which the distance reduces, such as when the actor wants to reach another point in the stage and

has to necessarily pass near the robot; by looking at the whole picture the general goal appears evident, even if at certain points collected values may hint at a different narrative.

Therefore, to prevent these issues, the computation of all the features described in chapter 2 should pass through a process of temporal smoothing, to make each classification derive not from puntual observations but from sequences of data collected in a certain time-span, to filter out noise and infer dynamics in a more robust, stable and accurate way.

In particular, the approach employed in implementing the framework's components that perform data collection and processing has been to organize collected data into **sliding windows**. Measurements are not handled as soon as they arrive but are instead stored in a buffer, which is emptied and processed periodically, based either on the number of collected points or on a fixed frequency. Data contained in each window are then smoothed using different techniques to assess the feature's value in the covered time period, which are explained in detail in each related section.

5.1.2. Handling of missing values

Given the fact the robot is deployed in the context of a theatre performance, it may happen that some input feature is not always available. This can happen, for every feature of interest, when the actor is not inside the receptive field of the sensor, with the pre-processing layers failing to detect the actor's presence and consequently not being able to compute any descriptor.

It's important to remark that this behaviour is not a failure of the data acquisition process, but there could be instead artistic reasons that make the actor undetectable by the robot. For instance, the actor may want to stay outside the field of view of the robot (*e.g. behind its back*) to convey a certain narrative, making the emotion-detection pipeline fail in detecting either facial or body expressions.

To handle these situations, fallback data points representing the absence of the actor are used, namely a point outside the stage for the proxemics analysis and a neutral emotion for the emotion classifier.

5.1.3. Handling of robot movements

Besides missing observations, another side effect of deploying the system on a physical robot is the impact of movements on acquired data. This impact can be of two types:

Misinterpretation of acquired data Considering again the example of proxemics, the actor-robot distance can reduce when the robot is moving in reply, even if the actor's intention is to keep it as large as possible. In this case, the movement of the robot can create a noticeable shift in the inputs' values, which is not directly linked to a shift in the actor's expressions or intentions.

Noise in the data Movement on the stage surface, and in particular the characteristic body shape and structure of the reference robot, can create wobbling and instability for instance in the camera sensor, leading to "non-deterministic" data fluctuations.

For this reason, the system should be aware of when the actuator system (considered as a third-party black-box system outside the scope of this work) is making the robot move, to ignore incoming inputs and to consider only those caused by actions of the human actor.

This mechanism is implemented by means of a feedback topic through which the actuator informs the system about the beginning and end of the robot's reply and movements. Upon receiving messages on this topic, the system toggles the state of the nodes connected to sensors, activating either the normal free state, where inputs are handled, or a blocked state, where inputs are ignored.

5.2. Remarks on image performance in ROS2

Another important remark that has to be done before describing the implementation of each component of the system is related to the acquisition and handling of visual inputs.

The basic and most widespread solution is to have a node connected to the camera acquiring frames and publishing them on a specific topic, which are then read by listener nodes and handled accordingly. Both ROS1 and ROS2 provide official pre-implemented packages to interface with USB cameras and publish frames at the desired framerate. A first implementation has been therefore made following this approach, using the official *usb_ cam* package to control the webcam and custom nodes to run inference on frames.

To test the inference computational footprint it has been measured the time elapsed between the availability of the image (*i.e. after the messages have been decoded and the image object created*) and the availability of inference results. During these tests, however, latency has been found to be noticeably higher with respect to that collected by profiling the algorithms outside the ROS environment, directly reading data from the camera.

This discrepancy led to a further investigation about the cause, which could be two-fold:

- impact of the ROS environment on the algorithm's performance, slowing it down due to thread-management and other low-level issues
- corruption of the image file during the encoding/decoding steps, that could alter its size and make it heavier to be processed

After several tests, the issue has been identified to be caused by memory management in ROS Python APIs, which makes every access to the frame (belonging to a different memory space) slower when compared to accesses to frames acquired in the same process (and hence directly available in the same memory space of the consumer algorithm).

A possible solution could therefore be to create a shared memory space across the camera controller and consumer nodes, to reduce the inter-process latency at the root of the issue. In the ROS1 framework this is made possible by a built-in package, whereas in ROS2 it is realized through the concept of *node composition*, which creates a single process composed of all the specified nodes. Such capability is however available only for nodes developed using C++, and Python nodes cannot be composed. This sets a strong limit to its adoption in this work, since the Python environment is necessary for the use of inference algorithms, and other solutions have therefore been explored, such as the built-in Python shared memory module or the Redis library, but none of them has shown a significant improvement over the baseline message-based implementation.

The final solution has eventually been to merge all the nodes requiring vision capabilities *(i.e. needing to access the camera feed)* into a single one, where the main thread is connected to the camera and consumer algorithms run inside internal threads, using buffers to share acquired frames.

This approach makes also sense from a conceptual point of view since it enables the reuse and sharing of partial results among nodes using the same input sources for inference, such as the emotion recognition from facial expressions and eye contact detectors, both exploiting the image of the actor's face. In this way, a single face detector is implemented, whose results are directly provided to the requiring algorithms.

Figure 5.1 shows the internal organization of the vision node, whereas figure 5.2 shows a sequence diagram describing how threads are activated during a single run of the image acquisition loop. For the sake of conciseness, thread names have been shortened.

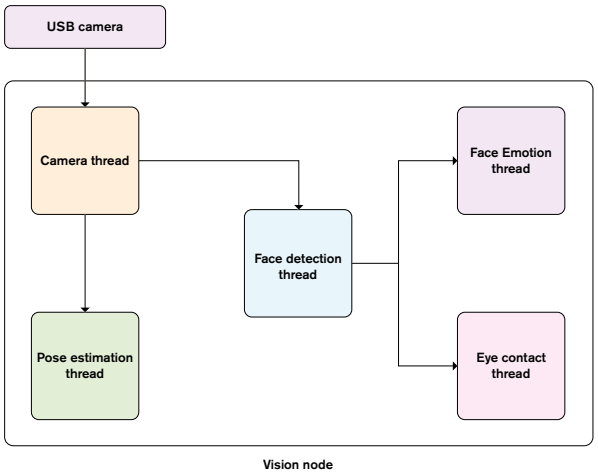


Figure 5.1: Block diagram showing the internal organization of threads in the vision macro-node, as well as data flows and hierarchy among them

5.3. Emotion recognition

To converge to an emotion-recognition model, several experiments have been conducted, in terms of both the model's algorithm and architecture, and input data and related pre-processing steps.

In light of the remarks made in 2.1 about how emotions are often expressed, the emotion recognition module is composed of two submodules, to characterize both facial expressions and whole-body movements.

To ensure coherent results, the two sub-systems share the same output space, using in particular the six basic emotions proposed by Ekman as possible labels. Outputs are encoded in a multi-dimensional vector, containing not just the characterization of the dominant observed emotion, but confidence values for all the possible states, to allow for future temporal smoothing and filtering steps, as shown in 5.1, where t is the current time step and i the recognition sub-module.

$$E_{i,t} = (an_{i,t}, di_{i,t}, fe_{i,t}, ha_{i,t}, ne_{i,t}, sa_{i,t}, su_{i,t}) \tag{5.1}$$

This formulation also enables vectorial operators such as sum and element-wise product, to update confidence values in parallel when computing the operations described later.

Figure 5.3 shows at a high level how emotional signals extracted by these sub-modules are integrated to generate a final prediction of the actor's emotional state. Vectors of confidence values are locally accumulated and smoothened over a time window of pre-defined

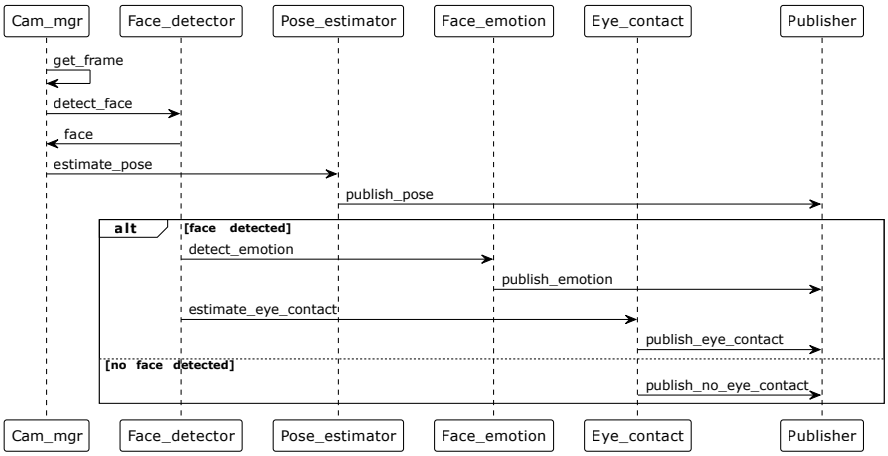


Figure 5.2: Sequence diagram showing the activation of threads in the vision macro-node inside the image acquisition loop

length (*i.e., 1 second*), to filter out noisy observations due to the model's inaccuracies or quality issues of input data, and eventually combined into a single output emotion.

It's worth to point out this architecture can be easily extended with new emotion-recognition modules defining proper inputs, aggregation and selection blocks.

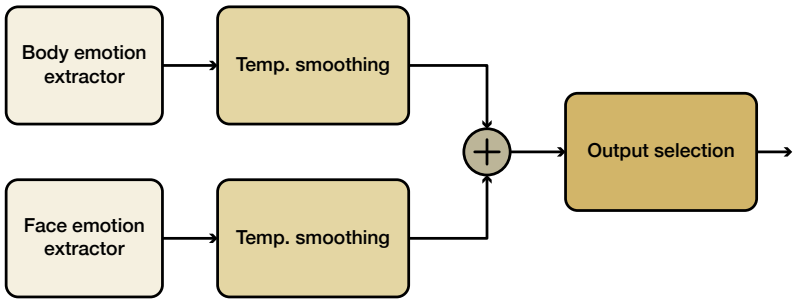


Figure 5.3: High level block diagram of the aggregation and final prediction of actor's emotional state from different specialized recognition modules

Smoothing has been implemented in 2 different ways: by computing a standard moving average of all the confidence values in a window of size M, given by equation 5.2

$$\bar{E}_{i,t} = \frac{E_{i,t} + E_{i,t-1} + \cdots + E_{i,M-(t-1)}}{M} \tag{5.2}$$

or by computing an exponential moving average, given by equation 5.3, which has a forgetting effect and weights differently the most recent data points according to the

parameter α, to be more reactive to changes in the expressed emotion.

$$\bar{E}_{i,t} = \begin{cases} E_{i,0} & \text{if } t = 0 \\ \alpha E_{i,t} + (1-\alpha)\bar{E}_{i,t-1} & \text{if } t > 0 \end{cases} \tag{5.3}$$

5.3.1. Emotion recognition from face

As described in the Ekman emotions model, facial expressions are considered to be a universal display of emotions, whose meaning is to a certain extent shared across different cultures and species. This universality is also used to back the hypothesis on emotions category, grounding the argument on the fact that the same facial muscles are used to display similar emotions in different cultures [92].

Due to this, several works and researches exist on the classification of emotional states from the analysis of facial signals [26], constituting more than 95% of the whole research body on the topic [30] and in recent years several big tech companies also launched their own commercial services and APIs offering face detection and face-based emotion recognition from pictures. This "race for the face", with facial recognition systems seeping into a wide variety of applications and their related social and legal implications, also caused political institutions to set regulations on their usage [5], to avoid their misuse in critical contexts and negative impacts on people's life, for instance discrimination or deception, due to algorithmic uncertainties, low-quality data and mindless reliance on the model's outputs.

Given this abundance of existing solutions and research projects on the field, and to avoid wasting energies for the development of a possibly sub-optimal classifier, this module has been implemented through third-party open-source libraries. Moreover, to address privacy concerns explained above and minimize latency, only pre-trained and on-device inference systems have been considered.

Two models have in particular been tested, both available as Python libraries: PAZ and DeepFace. The section is therefore organized as follows: first a brief introduction to each solution is given, analyzing the network architecture and example usage; then the implemented model is selected, providing motivations for the choice. Given the fact that both libraries rely on a face detection step prior to the inference, available face detection algorithms are then compared (on the basis of their latency/performance tradeoffs) for selection and eventually the deployment of the chosen one is described, along with implementation details to overcome its common technical criticalities.

PAZ

PAZ (acronym of *Perception for Autonomous Systems*) [7] is a software library developed by *Arriaga et al.* that provides perception and processing modules at different levels of abstraction, which can be combined in different topologies for several tasks, including emotion classification from face.

In particular, PAZ APIs feature three levels of abstraction which also correspond to three different levels of data manipulation, namely mapped onto the concepts of *pipelines, processors* and *backends*.

Emotion classification is offered as a *pipeline*, hence at the highest abstraction level, and inference can be performed with few lines of code, as shown in the example below:

Listing 5.1: Example of emotion inference on a single image using PAZ emotion classifier

```
from paz.pipelines import DetectMiniXceptionFER
detect = DetectMiniXceptionFER()
inferences = detect(image) # Image in numpy-format
```

The result of the method call is a dictionary, containing an array of detected faces, along with their dominant emotion and confidence value and an annotated image, showing all the bounding boxes and the corresponding prediction. However, this output format is not suitable to achieve the desired structure of the emotion classification vector described in 5.1, since only the dominant emotion is reported, and time spent to generate the summary image can be avoided since is not necessary for the intended purposes.

For this reason, a custom processing pipeline has been implemented, overriding internal classes to make the module output a vector of all the possible classes and related confidence values. In particular, a custom *processor* has been defined, which performs in sequence the following steps:

- **detection** of faces in the image
- **cropping** of the detected ROI to generate face patches
- **classification** of each face patch, using a convolutional neural network

eventually outputting a list of detected emotions and related scores.

In both cases, the classifier is internally implemented by means of MiniXceptionFER [6], a variation of Xception [23] trained on IMDB-gender and FER-2013 datasets, whose final architecture is a *"fully-convolutional neural network that contains 4 residual depth-wise separable convolutions where each convolution is followed by a batch normalization*

operation and a ReLU activation function", as shown in figure 5.4.

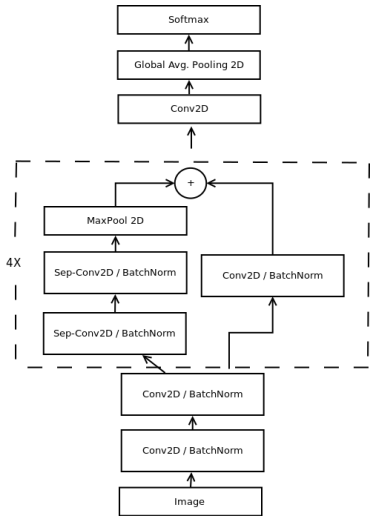


Figure 5.4: MiniXception network implemented in PAZ emotion classifier [6]

Inference latency of this model has been then measured, by acquiring a reference video with the webcam mounted on the robot and collecting a hundred measurements of the time spent to produce the result on a single frame, which has been rescaled from the original resolution of 1280×852 to a height of 480 pixels. In the end an average latency of **0.0254s** with a standard deviation of **0.00245** has been measured only for the emotion classification.

DeepFace

DeepFace is an open-source Python package wrapping several state-of-the-art models for face recognition and facial attribute analysis.

It features a transparent face recognition pipeline, composed of detection, alignment, normalization, representation and verification, on top of which several APIs are provided, including emotion classification.

Emotions can be in fact inferred from an input image with a single line of code, shown in the code snippet below:

```
from deepface import DeepFace
obj = DeepFace.analyse(img_path, actions = ['emotion'])
```

which returns a dictionary containing confidence values for all the six basic emotions, as well as a reference to the dominant one. For this reason, in contrast to PAZ, there is no need to define custom components for the post-processing of the model's output.

The face analysis module also offers the possibility to specify the desired face detection backend, used to identify the face ROI in the input image. Available backends are OpenCV (the default one), SSD, MTCNN, RetinaFace and Dlib, differing in terms of performance and speed. It's important however to note that cases in which a face is not present in the input image are handled in a tricky way; in case the enforce_detection parameter is set to True, a ValueError exception is raised, requiring the caller to properly handle it, whereas if it is set to False, confidence values are still returned, without any information whether a face is present or not.

The emotional expression analyser takes the name of *HyperExtended LightFace* [94, 95] and its emotional classifier is implemented as a custom-defined neural network of 12 layers (5 convolutional and 3 fully connected), outputting a probability distribution over the six Ekmann emotions and the neutral state.

Model selection

Both the PAZ and DeepFace models run on patches with a resolution of 48×48 pixels (the resolution of the FER-2013 dataset samples), and hence a resizing step is needed prior to feeding each input.

Selection between these two candidates has been made by considering their precision metrics, available in the corresponding papers [6, 95]. Even if scores are quite similar the **MiniXception** model used in PAZ performs, in general, better than the LightFace architecture employed in Deepface, and has hence been chosen.

Face detection

As stated in the introduction of the section, both tested models rely on a face detection step to crop a face patch from the original image, which is then resized to 48×48 pixels. Because of this, the choice of the face detector is independent from the choice of the facial emotion model, and state-of-the-art detectors have therefore been tested, being them OpenCV with a HAAR Cascade detector, SSD, Dlib, RetinaFace and MTCNN.

Selection has been in particular performed using the same reference video described above, measuring the inference latency on each frame on the robot's computer.

Table 5.1 reports the collected results and shows a huge gap in terms of latency among the available face detection techniques, offering different trade-offs between detection accuracy and inference time. These techniques in particular can fall into two large categories:

- Geometric approaches, such as OpenCV, SSD and the HOG + SVM version of

Detector backend	Average Latency (seconds)	Standard deviation
OpenCV	0.0121	0.0008
SSD	0.0177	0.0015
Dlib	0.0611	0.0013
RetinaFace	1.4755	0.106
MTCNN	0.2730	0.0201

Table 5.1: Summary table showing the collected performance metrics (*i.e. average inference latency in seconds and standard deviation*) for all the tested face detection algorithms

DLIB; being based on traditional image analysis and computer vision techniques, they are characterised by a low latency but may fail at detecting faces in case of occlusions or certain head orientations

- Deep neural network (DNN) approaches, which instead use a neural network trained on samples of faces to analyse each image and estimate the presence and location of a face, offering a more robust detection but at the cost of higher latency.

Because of the real-time requirements of the systems, and in order not to overload the computational resources of the on-board computer, only the geometric approaches have been considered (namely OpenCV, SSD and Dlib), since latencies of the DNN based ones would have created a bottleneck in the systems, capping its performance at less than 5 frames per second.

Several further tests have then been performed to understand in which critical situations each detector fails to recognize the actor's face. From these tests emerged that they all share the same criticalities, such as the person looking sideways and an head tilting of more than 45 degrees, as shown in figure 5.5, whereas no noticeable difference of accuracy has been found.

For this reason, having the lowest computational footprint, OpenCV has been selected.

OpenCV deployment

As described in 5.2, the face detection module is shared among the facial emotion- and eye contact- detectors and is hence necessary to implement a 2-steps pipeline, that first detects a face (made then available to other nodes) and subsequently computes the emotion, instead of an integrated end-to-end emotion classifier.

For this reason, after selecting OpenCV as the to-go face detection system, the final decision has been to implement it by directly using the OpenCv Python package. The OpenCV-backed DeepFace stand-alone detector is in fact just a wrapper for the vanilla

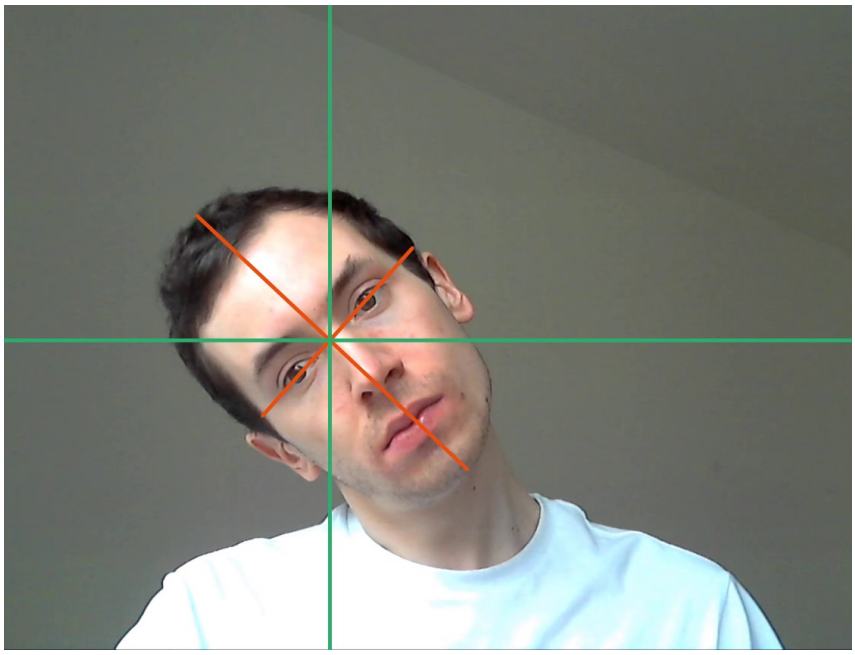


Figure 5.5: Orientation of the head (around 45 degrees) that makes OpenCV backend fail in detecting a face in the input image

implementation and wouldn't make sense to add additional dependencies and layers for the same functionalities.

Eventually, to overcome the limitations due to head tilting a common solution has been implemented and detection is repeated on the same image with multiple angles of rotations of the same image, namely 0, 45 and -45 degrees. To reduce the number of unsuccessful detection trials the implementation of this technique employs a caching mechanism, storing the rotation angle used at the previous step and selecting it as the first choice at the next one, based on the assumption that nearby frames will have similar values of head's orientation. In this way the overall accuracy of the model is increased without worsening the average latency, which does not differ much from that reported in table 5.1, making the system still able to fulfil the real-time detection requirements.

5.3.2. Emotion recognition from body

The second cue from which the emotional state of the actor is inferred is their body, in particular their posture and the quality of their movements.

This source of information has been neglected for a long time in the field of automated emotion recognition [30], however with no historical or social evidence. Postural information has in fact been claimed also by Darwin in [29] to be an emotional expression, based

on the observation that body language is an important component of human-human communication able to express emotions and to improve their perception, as also confirmed by other studies [8, 39, 109].

Moreover, keeping in mind the theatrical context of this thesis and the scope defined inSection 1.1.1, body posture and choreography are often employed to convey meaning at different levels of nuance, and exaggeration of movements in artificial agents has been shown to improve the interaction's engament and perceived value [43].

For this reason several experiments have been performed, to first define which features could be extracted from the actor's body and subsequently understand which could be relevant to discern their emotional state, training several classification models and evaluating their performances on different input spaces.

The goal of the process is in particular to select a classification model that, given information about the actor's body movements and posture, classifies the expressed emotion (choosing between the six defined by the selected Ekman model plus a neutral class).

The first step in achieving this goal has been the selection of a proper training dataset, providing samples of body expressions annotated with the corresponding emotion. Using this dataset several classification models and features encoding have been tested, to select the best performing architecture on the basis of the validation accuracy metric. Decision of which models to test has been in particular grounded on common techniques in gesture recognition or context-related techniques available in the literature.

The section goes through each of these steps: first the reference training dataset is described; then for each tested model motivations and comments on its performance are given, to eventually explain the final selection and related implementation details.

Reference dataset

As already mentioned, a preliminary step to all the trials has been the research of a proper reference dataset providing annotated examples that could fit in a theatrical context and take into account the considerations made so far.

The choice eventually fell on the *kinematic dataset of actors expressing emotion* developed by *Zhang et al.* [118], composed of 1402 sequences of movements, performed by 22 semi-professional actors *(50% male and 50% female)*, portraying different versions of the six basic emotions introduced by Ekmann *(anger, disgust, fear, happiness, sadness, surprise)* together with a neutral state. Sequences are encoded as BVH files (Biovision Hierarchy), a standard motion-capture format that provides both skeleton hierarchy and motion data.

Data, in particular, have been acquired using a wireless motion capture system with 17 sensors working at a frequency of 125Hz, whose placement is shown in figure 5.6 (a), and sequences have been performed in a square stage of $1m \times 1m$, shown in figure 5.6 (b). In the end, 72 body landmarks, listed in figure 5.19, are detected, and described by means of their 3D coordinates (encoded as relative displacement from the hip center) and rotation.

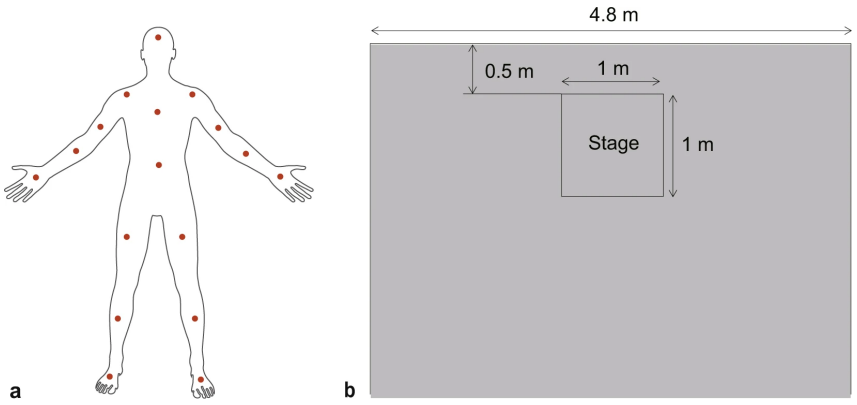


Figure 5.6: Sensor locations and laboratory environment [118]

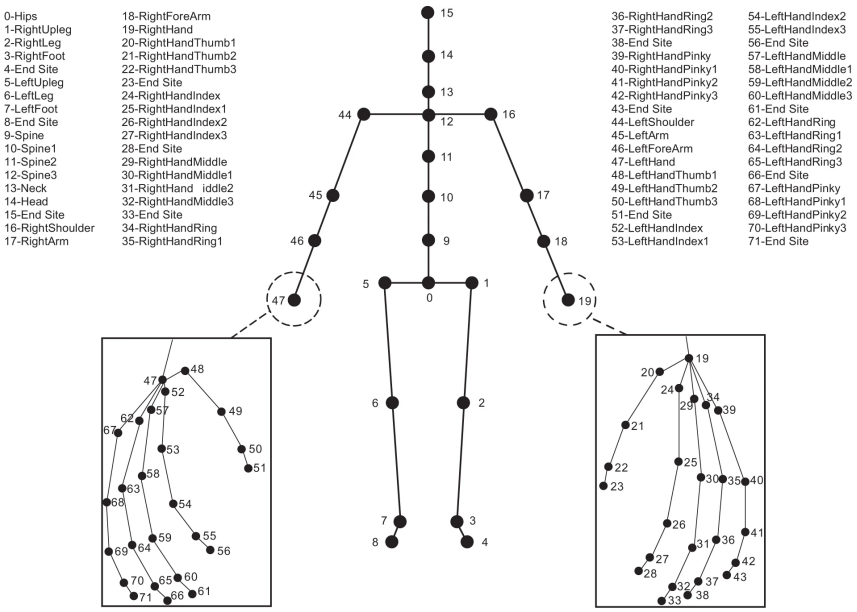


Figure 5.7: Skeletal structure and captured landmarks in the dataset [118]

To avoid repeating the computational cost of parsing and properly representing each BVH file present in the dataset, having also as a side-effect the reduction of the file size, a pre-processing step of converting the dataset into .csv dataframes has been performed (retaining only the relevant body joints, namely those detected by the pose estimation

module listed in table 5.6). Resulting files contain sequential information about all the relevant landmarks, having rows representing each time step and columns storing their coordinates, as shown in figure 5.8.

	LeftArm.X	LeftArm.Y	LeftArm.Z	LeftForeArm.X	LeftForeArm.Y	LeftForeArm.Z
0	-5.635052992821748	143.8687639530363	-9.56481039763968	-3.3191816876229088	116.98480745152402	-8.62161215625477
1	-5.635052992821748	143.8687639530363	-9.56481039763968	-3.3191816876229088	116.98480745152402	-8.62161215625477
2	-5.6351479928217465	143.86015795303632	-9.56422139763968	-3.319276687622907	116.97620145152403	-8.62102315625477
3	-5.647176084745686	143.85743299719007	-9.57215413905494	-3.3313059374417584	116.97347638986996	-8.628956070484172
4	-5.671338535103201	143.85923967593254	-9.589383044068065	-3.3554673000868807	116.9752831501306	-8.646185322688956
5	-5.6980558035280815	143.8613390052256	-9.619779614000052	-3.389529011897746	116.9772908953123	-8.661311421418143
6	-5.735366640944905	143.85891620274708	-9.657057589861086	-3.432694168823594	116.97492037342629	-8.683164507846342
7	-3.940345843849777	143.16485594656532	-11.002438551750922	-2.6726644581236583	116.34406774774564	-8.167248205700337
8	-3.99991031250587	143.1745417239252	-11.05368895039812	-2.551644320661325	116.36722063653436	-8.177917447931828
9	-4.067658168119133	143.19116038468275	-11.076352988642768	-2.511335452818762	116.3904339410888	-8.19556807798348

Figure 5.8: Excerpt from a converted .CSV file of the dataset, showing row-column organization of skeletal sequences

At the time of writing, no performance baseline is provided in relation to this dataset and hence no comparison could be made for all the tested models.

Training has in particular been performed on sequences of motion descriptors (either spatial/kinematic or LMA-inspired), to take into account the considerations made so far about how a movement expresses a content at lower frequencies than those at which data are collected. For this reason all the tested models are based on recurrent architectures, differing either for the kind of features provided as input or for the architecture itself, which outputs a probability distribution over the seven available emotion classes, passed as-is to the emotion fusion step.

The first set of evaluated models are based on an LSTM and a fully-connected part, whereas the last experiment asseses the feasibility of a recurrent graph neural network for body emotion classification. Models, moreover, have been trained on multiple lengths of the data sequences, for which the temporal evolutions of loss and accuracy (both in training and validation stage) have been collected and reported in comparative charts.

Experiments with sequences of landmark coordinates

This first experiment has been designed to provide a baseline for the evaluation of the other considered solutions. It is in particular based on the sheer spatial information of the body landmarks, namely their coordinates on a bidimensional vertical plane.

As already mentioned the network uses sequences of data, for which different lengths are tested. For this reason, the parsed pose sequences are split into chunks of the desired length, to which is assigned the emotion label corresponding to the file (all sequences

deriving from the same pose sequence hence have the same attached emotion).

Model architecture The architecture of the model is the result of an iterative process, starting with a simple recurrent architecture for a classification problem to which complexity has been increasingly added until a change in performance has been detected.

After some tweaks, the best performance has been obtained with the architecture reported in figure 5.9, which is also used by all the other tested models (except for the graph-network-based one). It is in particular composed of an LSTM layer that takes as input a sequence and produces an embedding passed to the fully-connected part, composed of two dense and an output softmax layers. In order to improve generalization capabilities and accuracy on unseen samples, dropout layers are placed before the two dense layers.

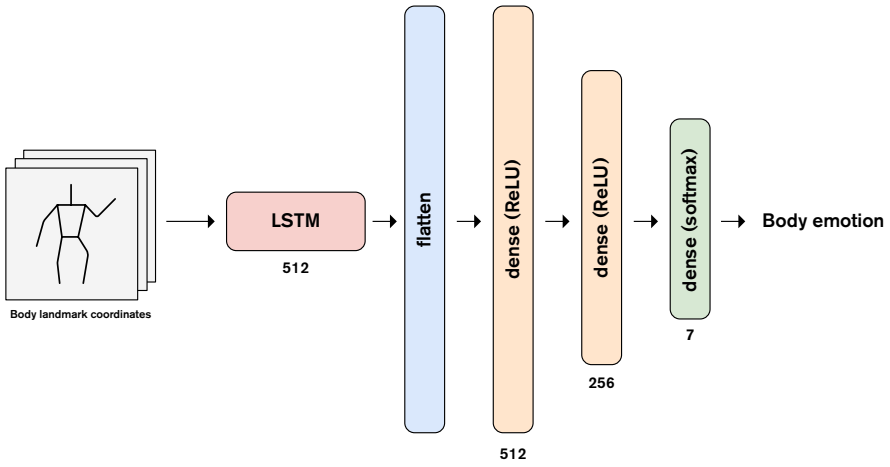


Figure 5.9: Diagram showing the architecture of the neural network classifying sequences of landmarks' coordinates into an emotion class

Performance considerations As already mentioned, the performance of the network (namely the accuracy on the validation set) has been computed over several lengths of the input sequences. For each length, the model has been trained for around 300 epochs using the RMSProp optimizer, with a learning rate of 0.0001 (higher values have been found to cause overfitting after few epochs) and a batch size of 32. Being the problem a multi-class classification problem, the Categorical Cross Entropy loss function has been used. Performance, eventually, has been assessed by splitting the dataset into training and validation sets, using a stratified splitting with a ratio of 0.25.

Graphs reported in figure 5.10 show, for each sequence length, the collected time evolution of loss and accuracy in both training and validation stages. From them, it can be observed

that the network performs really well with sequences up to a length of 3 poses, after which starts to exhibit an overfitting behaviour and a suboptimal accuracy on the validation set. In particular, the network reaches a validation accuracy on the validation set of $\sim$**0.96** on 1-frame sequences and of $\sim$**0.93** on 3-frames sequences.

Experiments with LMA features

In light of the considerations made in 2.2.1 about the ability of Laban Movement Analysis to describe both the movement's physical and semantic qualities, an experiment has been conducted to check whether LMA-inspired descriptors could provide the same (or even better) results as the coordinate-based one.

Input representation For this purpose, motion capture data have been pre-processed to extract LMA-based descriptors, inspired by formulas used in [62, 104].

Computed descriptors are, in particular:

- Module of jerk (third order derivative) of the left and right wrists, representing the **flow** component
- Speed and acceleration on the vertical y axis of the left and right hips, representing the **weight** component
- Contraction index of the body, computed using the formula in equation 5.4, representing the **shape flow** sub-component of shape
- Amplitude of the body on both the x and y axes (namely distance between max and min coordinate on each axis), related to the **shaping** sub-component of shape
- Dissymmetry index of the body, computed using the formula in equation 5.5 where $d_{left/right,center}(t)$ is the distance of the left/right hand from the trunk at time t, and distance betwen the shoulder and the wrist for both the left and right side. Both descriptors are related to the **body** component

$$C(t) = \frac{||P_{hip\,center,t} - P_{left\,hand,t}|| + ||P_{hip\,center,t} - P_{right\,hand,t}||}{2} \tag{5.4}$$

$$Dys(t) = \frac{d_{left,center}(t)}{d_{left,center}(t) + d_{right,center}(t)} \tag{5.5}$$

It's important to point out that even if [104] uses tri-dimensional coordinates (hence values on the z axis) to compute these descriptors, no considerable difference of performance has

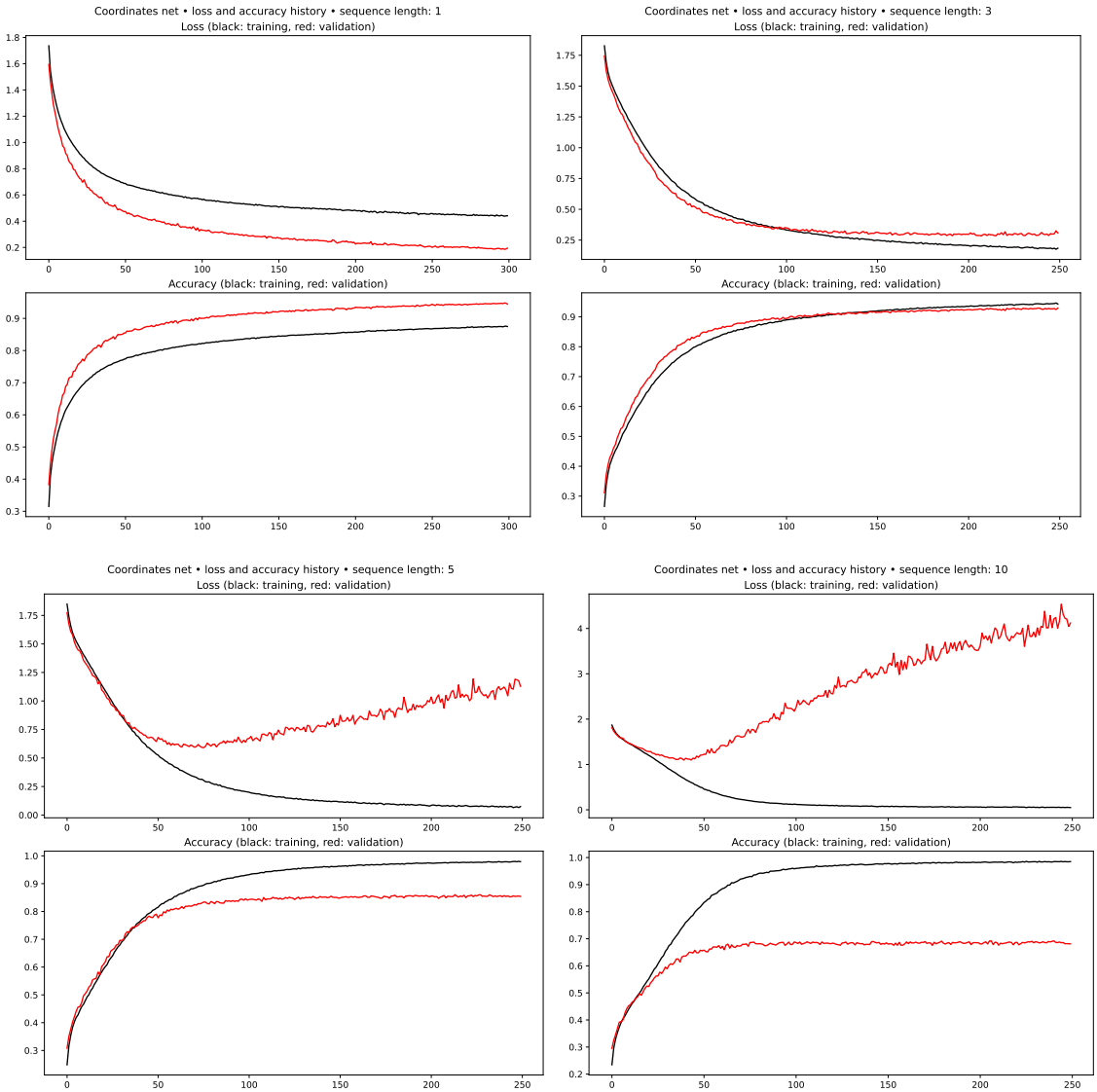


Figure 5.10: Plots of loss and accuracy of the trained networks in both training and validation stages for different sequences of body landmarks' coordinates

been observed between training the network with 2D or 3D poses, and hence for the reasons that will be explained in 5.4.1 the bi-dimensional version has been selected.

Model architecture As before, the architecture is retained from the original coordinate-based experiment, with the only difference being the kind of features given as input.

Performance considerations Plots of loss and accuracy over the training epochs, reported in figure 5.11, show a behaviour similar to previous cases, with only small lengths of sequences (namely 1 and 3) making the model perform in an acceptable way, while others lead to overfitting after few epochs. Besides the four lengths of sequences used in the other experiments, two additional lengths have been tested, to check whether a larger time scale would make Laban features bring more informative content with respect to plain coordinates, but without success.

Another remark that can be made about this solution is the fact that several of the LMA descriptors listed above are computed as derivatives (up to the third order) of the landmarks' coordinates. Because of this, input of the network is based on a larger window than that needed by the coordinate-based network, namely larger of at least four elements, since jerk is computed with a 2-steps look-ahead in time. For this reason, the network can sometimes exhibit a delayed behaviour, reacting to a certain sample only after other poses have been collected and all the descriptors computed, even though it shouldn't be a problem, given the low latency of pose extraction methods, that will be described in 5.4.

However, as it can still be seen from the plots, the network is not able to achieve accuracies higher than those of 0.96 and 0.93 scored by the coordinate-based one.

Experiments with sequences of velocities

The third experiment makes use of kinematic descriptors and is based on the observation that speed and acceleration of body parts are often linked to the internal emotional state. Speed and acceleration are, moreover, computed as a transformation of the landmarks' coordinates used in the previous model, and for this reason the leading hypothesis has been that they could provide similar (or even better) results.

Input representation Input sequences have in particular been processed to extract the instantaneous velocity of each joint k at every time step t_i, using the formula proposed in [62] and reported in equation 5.6, where $x^k(t_i)$ and $y^k(t_i)$ are the x and y coordinates of

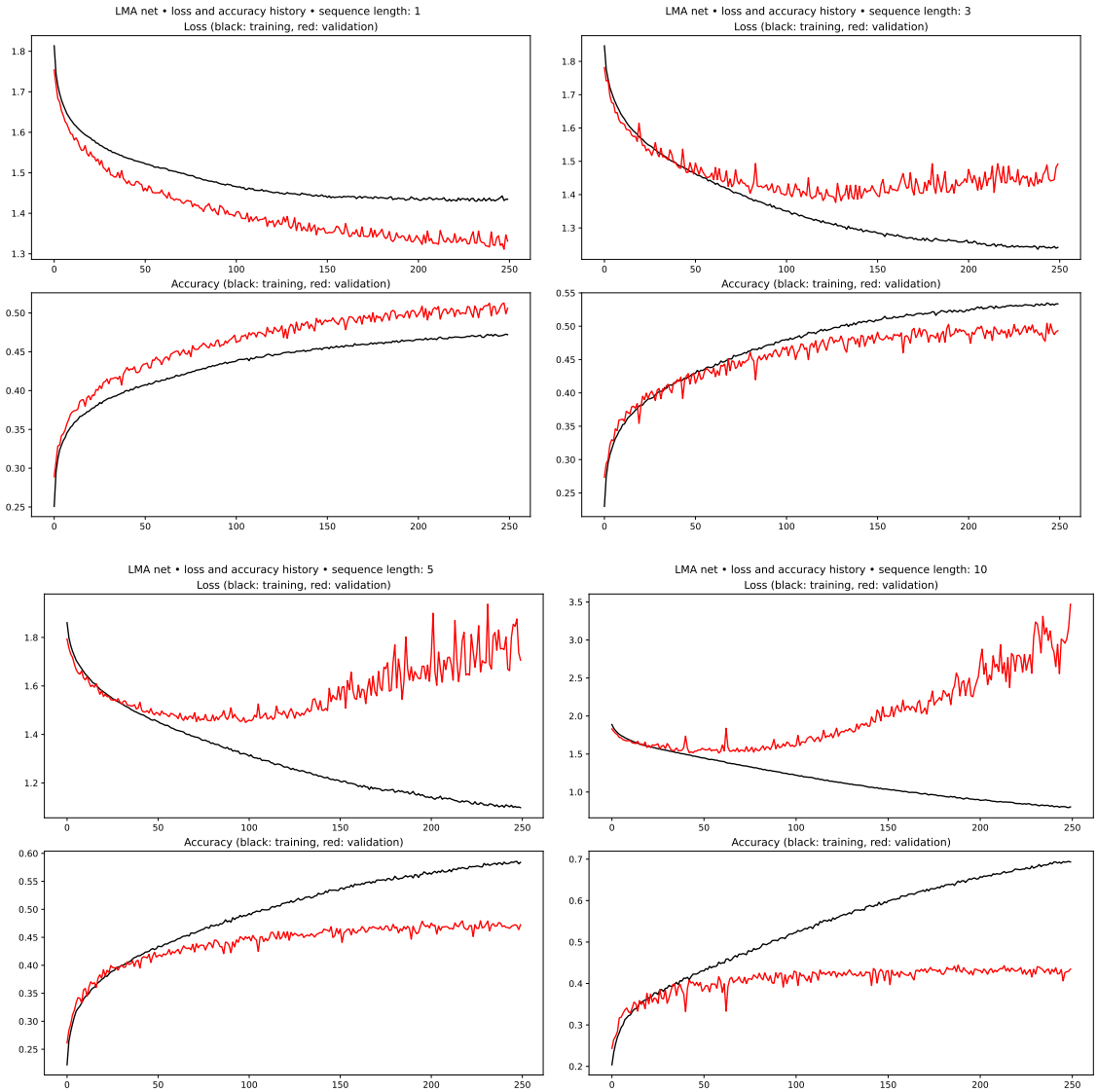


Figure 5.11: Plots of loss and accuracy of the trained networks in both training and validation stages for different sequences of LMA-based descriptors of movement

joint k at time t_i.

$$v_x^k(t_i) = \frac{x^k(t_{i+1}) - x^k(t_{i-1})}{2\delta t},\ \delta t = 1$$
$$v_y^k(t_i) = \frac{y^k(t_{i+1}) - y^k(t_{i-1})}{2\delta t},\ \delta t = 1 \tag{5.6}$$

Model architecture The architecture is the same of the previous model, since there is no difference in the input shape (being the new data only a time difference of previous ones). The same diagram shown in figure 5.9 hence applies also to this case, with the due modifications about data given in input to the model.

Performance considerations As in the previous case, the time evolution of loss and accuracy for both training and validation stages are reported in figure 5.12. From the plots, it can be observed that, apart from 1-frame sequences, the network overifts even after few epochs and, when this does not happen, the validation accuracy reaches anyway a lower value compared to the result of the previous experiment. This approach has therefore been discarded.

Experiments with graph neural networks

This last experiment uses geometric information of the displacement of landmarks in the view plane, and is motivated by the peculiar problem structure.

The human skeleton seen in figure 5.7, composed of a set of landmarks connected to each other by bones, can in fact be modelled as a graph, whose nodes are the landmarks and whose edges are the bones. Moreover, from a kinematic point of view, it makes sense to consider the dynamics of each landmark as influenced by its peer joints.

Theoretical background Graph neural networks (GNNs) are a special kind of neural networks dealing with graph representations of input data, and can be described as *"an optimizable transformation on all attributes of the graph (nodes, edges, global-context) that preserves graph symmetries (permutation invariances)"* [89]. They can be used for several types of tasks:

- Graph-level prediction, to assign a label to the entire graph as a whole
- Node-level prediction, to assign a label to a single node, according to its role within

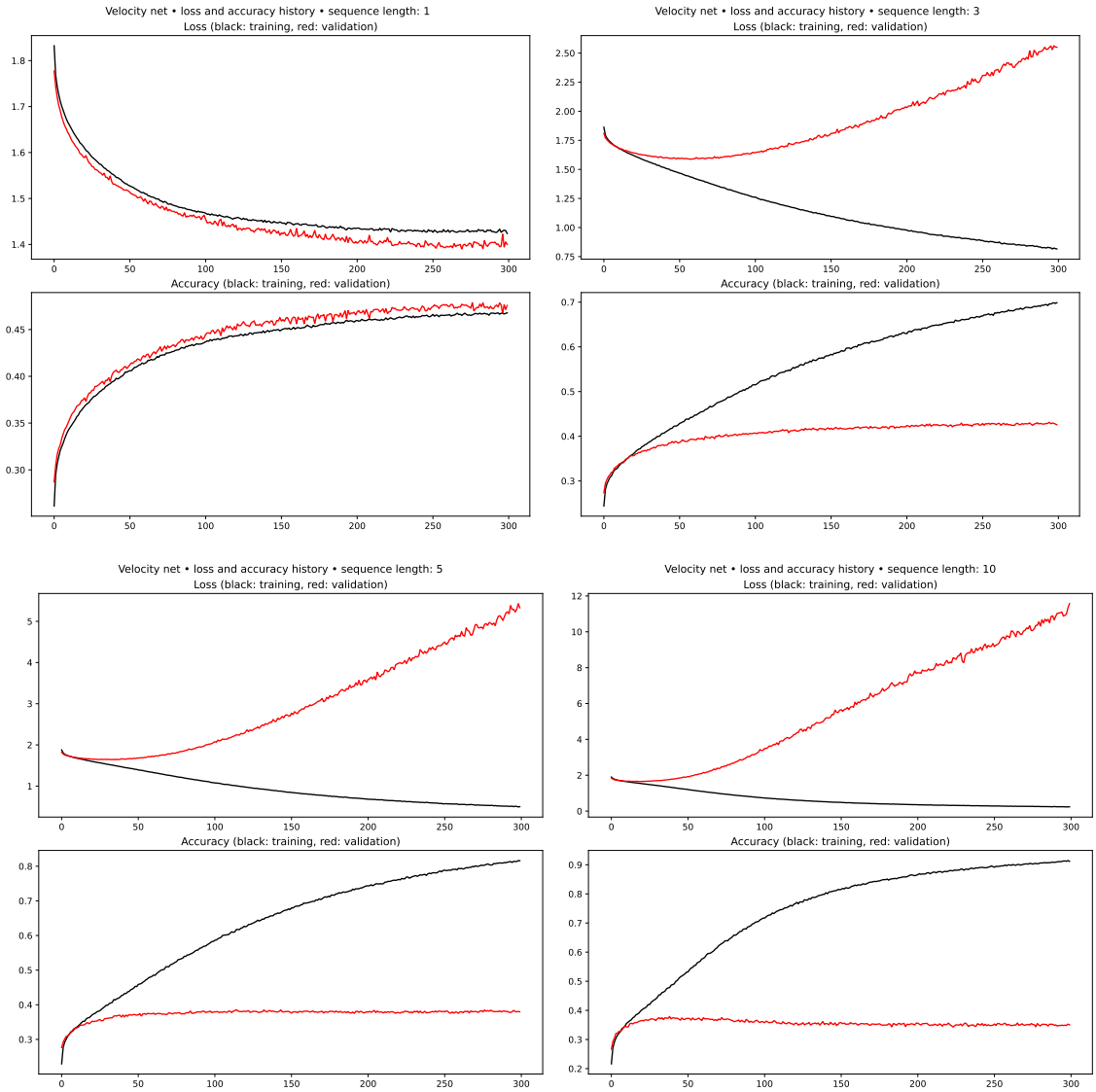


Figure 5.12: Plots of loss and accuracy of the trained networks in both training and validation stages for different sequences of body landmarks' velocities

the graph (*e.g. Karate club problem [116]*)

- Edge-level prediction, to assign labels to edges connecting two nodes, to predict for instance the relationship between two entities (*e.g. similar to a segmentation problem in convolutional neural networks*)

GNNs are also often associated to the concept of message-passing, first introduced in [44], a technique used to represent and implement interdependencies and mutual influences of nodes. Each node is in fact characterised by a hidden state, corresponding to the computed feature vector, which is updated at each turn with a function of the previous state and aggregated messages from neighbouring nodes.

Used frameworks and libraries Graph neural networks are supported by both the PyTorch and the Tensorflow framework, through several libraries. In this context, the PyTorch framework, paired with the PyTorch Geometric [38] and PyTorch Geometric Temporal [85] libraries, has been used to implement the tested network.

Input representation As stated at the beginning of the section, the network uses geometric information about the location of body landmarks in the bi-dimensional view plane. For this reason, each pose is taken as-is from the pose estimation step and converted to an (undirected) graph representation, using the data structures provided by the *PyTorch Geometric* library. The graph structure trivially resembles the skeleton model used by the pose estimation module and is therefore not shown.

In this implementation graphs are represented by two matrices:

- a feature matrix, containing the features of each node in the graph
- an edge index, describing how nodes are connected by edges

The feature matrix is in particular defined as a 13×2 matrix, containing for each joint (rows) its coordinate pair (columns).

Model architecture The implemented network is composed of two parts:

- a recurrent module, performing temporal graph convolutions on the given data sequences to update an internal hidden state
- a fully-connected (FC) classifier, using the computed embeddings to output a probability distribution over the emotion classes

which can be easily replaced with different implementations (regarding in particular the recurrent subnet) thanks to the modularity of the PyTorch framework.

The recurrent part has been in particular implemented using the GConvLSTM layer, based on the *Chebyshev Graph Convolutional Long Short Term Memory Cell* introduced in [93], which is the graph counterpart of the LSTM layer used in the other experiments, The fully-connected subnet is instead the same as in the other experiments, with two dense layers using ReLU activation functions intertwined by dropout layers and mapped to the final probability distribution by an output softmax layer.

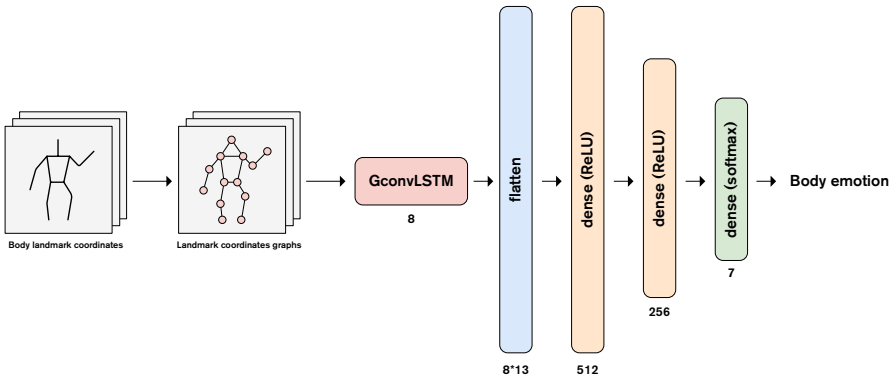


Figure 5.13: Diagram showing the architecture of the graph neural network classifying sequences of undirected graphs (encoding landmarks' coordinates and body joints relationships) into an emotion class

Performance considerations The model has been trained with the same approach adopted for other experiments, mapping the used methods to the PyTorch syntax. Results reported in figure 5.14 show behaviours that do not differ much from those of other networks not using a graph representation. In particular, the network shows a reasonable behaviour only with sequences of a single element, and an overfitting pattern appears for longer inputs.

Selected body emotion classifier

In light of results and considerations reported in the previous sections, the recurrent network based on body landmarks' coordinates has been selected as the implemented emotion classifier, because of its accuracy score, which outperforms by far the other tested solutions.

Choice of the input sequences' length, among 1 and 3 *(i.e. the best-performing versions)*, is instead based on empirical tests of both models in a real setting, deployed on the main computational unit of the reference robot and connected to poses directly estimated from the on-board camera (using the pose estimation that will be described in 5.4). By

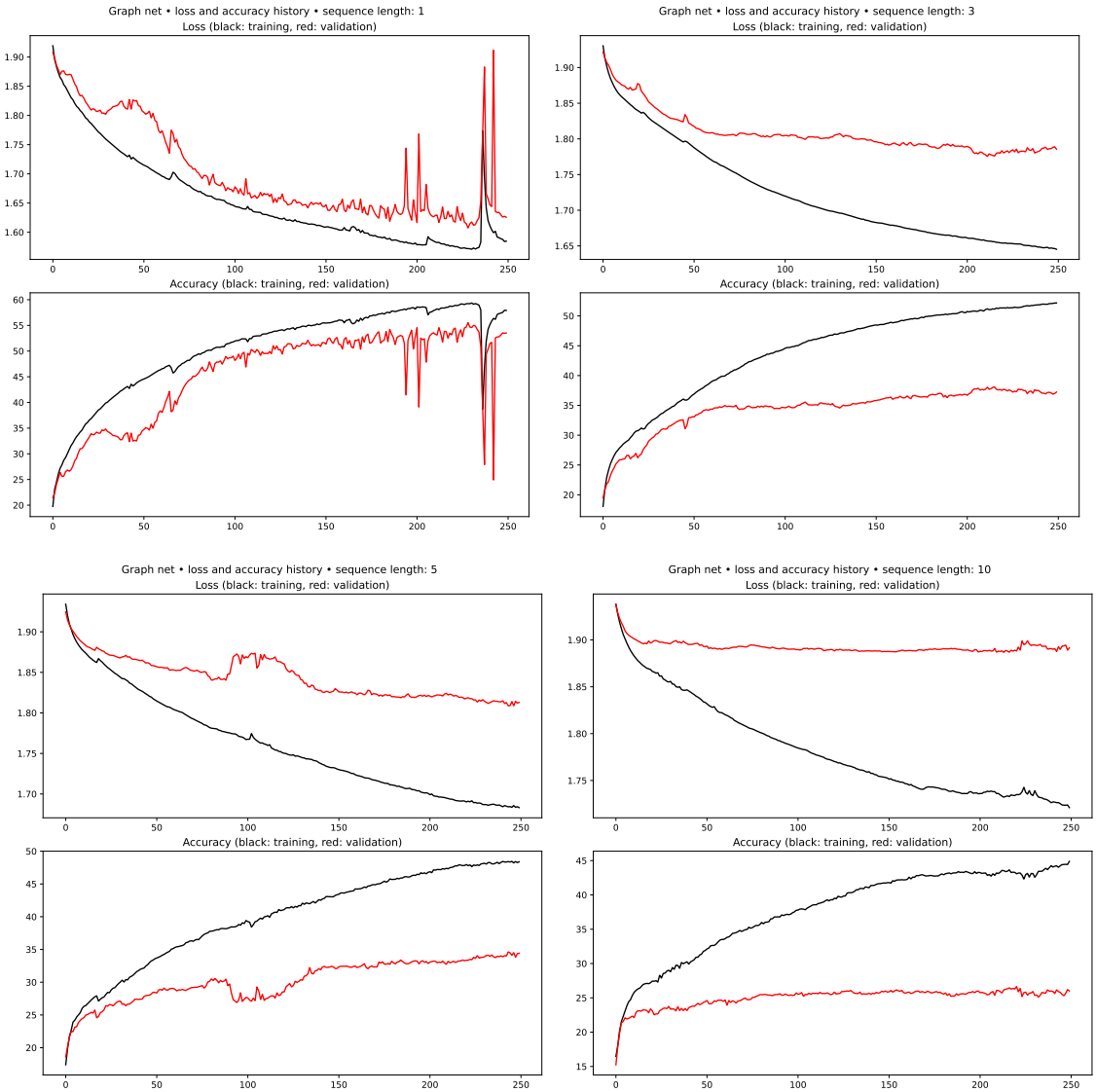


Figure 5.14: Plots of loss and accuracy of the trained networks in both training and validation stages for different sequences of LMA-based descriptors of movement

classifying in real-time the emotion expressed by a person moving in front of the robot, and by later analyzing the variability of those estimations, the model using sequences of 3 poses has been shown to yield more robust and accurate predictions than the one working with single snapshots, and has therefore been selected for *"in-the-wild"* inference.

5.3.3. Emotion fusion

After developing independently the two models to recognize the emotions expressed either through body movements or facial expressions, a fusion mechanism has been implemented, to yield a single resulting emotion $\hat{E}$.

This merge point has been conceived to provide a single characterization that could encompass all the ways in which the actor can express emotions, and has been therefore designed to be extensible in the future with an arbitrary number of emotion sources *(i.e. related to the other semantic layers described in 1.1)* and different fusion mechanisms, which can be implemented by specializing the *EmotionFusionMethod* abstract class.

The placing of this fusion point, namely at the end of the two processing paths handling facial and body cues (which are therefore kept independent and not correlated), is an example of *late-integration* of modalities, and has been chosen since it is the common way of integrating different modalities into a single representation, especially when dealing with gestures [113]. This approach is in fact usually preferred over an early merging right after feature extraction because it provides greater flexibility in modeling each processing path and avoids issues of high-dimensionality in their training sets [48]. However, as it can still be read in [48], there is not a general consensus over the optimal strategy for modality fusion, and several approaches have been proposed.

In the current implementation of the framework, the actor's emotion is inferred from their facial expressions and body gestures. Research on emotional expression and empirical tests have shown how these two sources can sometimes portray different (and even contrasting) emotions. At the same time, however, facial expressions have been shown by several works to be a more reliable and informative source with respect to body gestures and, because of this, emotions guessed from them are weighted more.

For this reason, emotion fusion in the system has been eventually implemented through this weighted mechanism, exploiting the vectorial representation of confidence values with the formula in equation 5.7, whose weights are inspired by the ratio proposed in [48].

$$\hat{E} = argmax\left\{E_{face} \cdot w_{face} + E_{body} \cdot w_{body}\right\}, \quad w_{face} = 0.7, w_{body} = 0.3 \tag{5.7}$$

For the sake of completeness, two other methods have been implemented in the system, since they are usually mentioned as possible solutions in the literature too [48].

They are in particular the element-wise sum and product of the two confidence vectors, whose formulas are shown namely in equation 5.8 and 5.9

$$\hat{E} = argmax\{E_{face} + E_{body}\} \tag{5.8}$$

$$\hat{E} = argmax\{E_{face} \odot E_{body}\} \tag{5.9}$$

5.3.4. ROS nodes

Regarding the ROS implementation, emotion recognition and fusion is realized by 3 nodes:

- *Vision* node, gathering frames from the camera and running the facial-expressions-based emotion detector as explained in Section 5.2
- *Body emotion* node, wrapping the neural network described inSection 5.3.2 to classify the emotion expressed through body movements
- *Emotion aggregation* node, connected to the other two and wrapping the emotion fusion mechanism described in Section 5.3.3

Emotion data, used either as partial results in the face and body processing paths or as the output of the emotion aggregator, are represented by messages containing the confidence values assigned to each detected emotion as well as a textual identifier of the publisher node, as shown in listing 5.2.

Listing 5.2: Format of messages describing the emotions detected by implemented emotion sources

```
float64 anger
float64 disgust
float64 fear
float64 happiness
float64 sadness
float64 surprise
float64 neutral
std_msgs/String identifier
```

5.4. Human pose estimation

As explained in 2.2, one of the features considered by the framework is the movement of actors, both in terms of posture and dynamic spatial qualities, and the emotion classifier

from body described in 5.3.2 has been trained on the coordinates of body landmarks.

For this reason one of the core parts of the framework is the estimation of the actor's pose, a common technique in computer vision aimed at estimating the spatial locations of body joints and landmarks of a person, given an image or a video feed.

As described in [28], early methods were based on hand-crafted features, defining metrics for matching based on structural representations of the human body, but encountered issues under visual occlusion or certain lighting conditions due to their lack of expressiveness. To overcome these limitations deep learning techniques have been applied to the problem, achieving remarkable results and becoming the state-of-the-art approach.

Among the several available solutions, either proposed in research papers or available as commercial products, different networks have been tested to select the optimal one for the actor pose extraction stage. In the following sections a brief description of every tested model is given, explaining the benefits and shortcomings in the particular context, as well as reasons for the final implementation decision.

Regarding this last point, and in light of the implementation in a theatrical context and near real-time processing requirements, evaluation was mostly based on the network's prediction smoothness across time (i.e. low jittering of predicted landmarks) and its computation footprint. Smoothness has been in particular evaluated by applying each model on a reference video of a still subject (standing still in the center of the frame) and computing the standard deviation of the estimated x and y coordinates of the left shoulder and hip (being them relevant joints for both the upper and lower body parts).

5.4.1. 2D or not 2D? Preliminary notes on representation

A preliminary step, however, has been to select which family of pose estimation methods to consider, and in particular which pose representation to use.

Human pose estimation (HPE) can in fact be divided into two macro-categories:

- *2D HPE*, aimed at estimating the body configuration typically from a single monocular image, retreving the landmarks' coordinate on a bidimensional plane
- *3D HPE*, placing estimated body landmarks in a tri-dimensional space, usually employing multiple monocular or stereo cameras

These two approaches have been evaluated, considering potential benefits and drawbacks in the considered theatrical context and in light of their future implementation on a moving mobile robot.

2D HPE

Bi-dimensional human pose estimation aims at localizing anatomical keypoints from a processed camera image of a person, providing information about their location in a planar cartesian coordinate system. Current state-of-the-art models are based on deep convolutional neural networks. This approach has been used for the first time in [103] and marked a shift from classical methods, solving several problems with limb-occlusion and improving accuracy and generalization capabilities.

As previously mentioned, 2D pose estimation computes the landmarks' location in a space relative to the given frame, assigning as coordinates the pixels in which the body part is estimated to be. For this reason, the size of the input image has to be taken into account and a normalization step is often needed to make predictions invariant to scale, especially when computing body movements such as in this case.

3D HPE

Tri-dimensional human pose estimation adds a depth dimension to the planar estimation and can be mainly achieved in two different ways: directly using a sensor providing depth information or reconstructing the 3D pose from a bidimensional image.

Differently from the previous case, the actual position of the person in the space is here provided, and estimated landmarks' locations are represented as world coordinates.

To implement a direct 3D pose estimation several sensors are commercially available, such as Microsoft's Kinect or Intel's RealSense. This approach, however, is not feasible to be implemented in this work, due to the instability brought along by a moving robot. Oscillations of the robot's body, on which the sensor would be mounted, add in fact noise to the observations, causing issues with calibration and the laser tracking algorithm, eventually making the sensor provide results not good enough to be used for computing all the needed features.

The second solution is the estimation of a tri-dimensional pose from a bi-dimensional image, employing a deep neural network to estimate the depth of detected body landmarks in a process also called *3D lifting* [100, 111, 115]. However, as stated in [70], *"the lack of 3d ground truth posture data for images in the wild makes the task of inferring 3d poses directly from colour images challenging"*.

This issue is particularly relevant in the context of this work, since actor's poses are not always similar to prototypical configurations often found in everyday life and available datasets, making the models estimate semantically wrong poses.

Conclusion

In light of all the considerations reported above 3D pose estimation has been discarded, because both the available solutions were not capable of offering a reliable and meaningful detection. Moreover, as it will be described in 5.3.2, 3D representation does not improve the implemented emotion recognition model in a considerable way, achieving a performance score extremely close to the one trained using 2D poses. 2D pose estimation has therefore been selected as the source for the actor's body pose, adopting the skeleton model for its representation. In the following sections available 2D HPE state-of-the-art models are analysed, to eventually define the implemented pose estimator.

5.4.2. Detectron 2

Detectron2 [114] is a Facebook AI Research's project that covers several computer vision applications, based on the PyTorch framework and available as a Python library. Supported pre-trained networks are available through its model zoo, where solutions for multi-person pose estimation are provided, offering different trade-offs in terms of inference speed and accuracy, as shown in table 5.2.

Model name	Inference time (s/image)	Average precision (%)
R50-FPN	0.066	65.5
R101-FPN	0.076	66.1
X101-FPN	0.121	66.0

Table 5.2: Table showing pre-trained body keypoint detection models (based on keypoint R-CNN) and corresponding baselines available in the Detectron2 library

All the models output 17 body key points, using the COCO format shown in table 5.3, and is possible to additionally retrieve segmentation masks. Each key point is characterised by 3 values: the x and y location in the image and a confidence score.

After some tests, this network has however shown some problems with "uncommon" poses (that could anyway occur during a theatre/dance performance) or with occluded body parts, leading to incorrect results, especially if more than one person (or misclassified person-like objects) are detected and overlap in the view field. Moreover, the range of confidence score for keypoints is not well-documented and there is no direct way to determine whether a keypoint is visible or not.

From the stability analysis of the model on the reference video, whose results are reported in figure 5.15, it can be observed that the model is not able to keep a smooth prediction

Index	Key point
0	Nose
1	Left eye
2	Right eye
3	Left ear
4	Right ear
5	Left shoulder
6	Right shoulder
7	Left elbow
8	Right elbow
9	Left wrist
10	Right wrist
11	Left hip
12	Right hip
13	Left knee
14	Right knee
15	Left ankle
16	Right ankle

Table 5.3: Table showing the COCO encoding of keypoints used by pose estimation models available in the Detectron2 library

across frames, with some landmarks disappearing and reappearing. This jittering would also cause problems in the processing of the extracted pose, such as estimating the speed and acceleration of body joints, where this kind of noise could lead to incorrect results.

5.4.3. OpenPifPaf

OpenPifPaf [59] is a library that performs both pose detection and tracking, employing a neural network architecture that uses Composite Fields to detect and construct a spatio-temporal pose: a single, connected graph whose nodes are the semantic keypoints in multiple frames. The architecture is based on either ResNet50 or ShuffleNetV2 for the encoder, to which are attached convolution models to compute composite fields, which are then converted into pose estimates by the decoder module. Each keypoint is characterised by a confidence score, a real-valued coordinate pair and a size estimate.

Table 5.4 lists pre-trained models available in the OpenPifPaf library, showing for each the corresponding performance on a NVIDIA GTX1080Ti GPU, as reported in the official documentation.

From these values, the *ShuffleNetV2k16* model has been chosen as the implementation candidate, since it offers a good tradeoff between precision and latency.

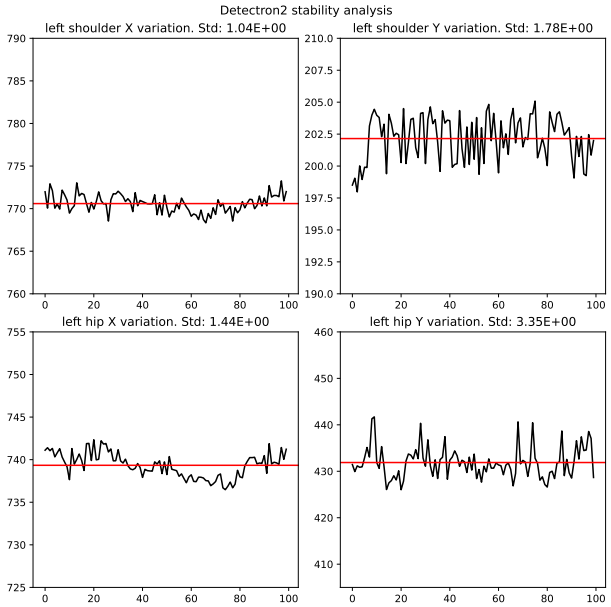


Figure 5.15: Stability analysis of the x and y coordinates of the left shoulder and left hip landmarks estimated using Detectron2 on a still subject

Model name	Inference time (ms)	Average precision (%)
resnet50	53	68.1
shufflenetv2k16	40	68.1
shufflenetv2k30	81	71.8
mobilenetv3small	26	47.1

Table 5.4: Table showing pre-trained pose estimation models and corresponding performance metrics available in the OpenPifPaf library

Outputs of the model are formatted using the same COCO encoding used by Detectron2 and reported in table 5.3. From the results of the stability analysis, reported in figure 5.16, it can be observed that the model still suffers from a a high jittering among consecutive frames, leading to the same issues described in 5.4.2 for Detectron2.

5.4.4. OpenPose

OpenPose [21] is a library wrapping a network able to perform real-time multi-person detection and owes its popularity to having been the first model able to jointly detect human body, facial and foot keypoints. It provides two output formats for the pose:

- COCO, containing 17 landmarks as described in 5.4.2
- COCO 25, containing 25 landmarks, adding hip center and feet keypoints

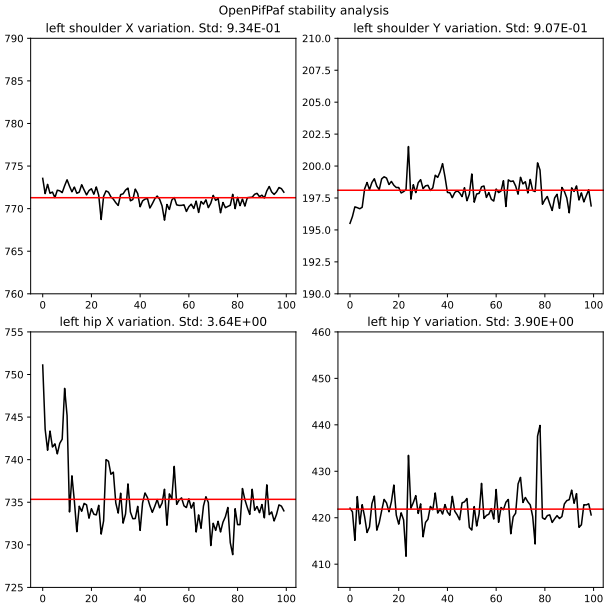


Figure 5.16: Stability analysis of the x and y coordinates of the left shoulder and left hip landmarks estimated using OpenPifPaf on a still subject

The library is written in C++ and based on the Caffe framework [54]. It requires by default a CUDA-capable machine, which is not the case for the target deployment machine. It can be nevertheless recompiled to remove CUDA dependencies and support cpu-only architectures, but doing so significantly reduces the network's frame-rate. For this reason, this solution has been discarded and no further analysis has been conducted.

5.4.5. Google PoseNet

Another tested solution was the use of the Coral Edge TPU platform, in particular the Coral USB accelerator depicted in figure 5.17 running the PoseNet model.

The Coral USB accelerator is part of the Coral Edge TPU (*Tensorflow Processing Unit*) family, a set of neural networks hardware accelerators to power on-device AI on embedded devices, to enable smart capabilities in resource-constrained settings without relying on cloud AI and ML services that could bring along potential privacy issues and concerns. This solution has therefore the additional benefit of delegating the inference part to an external component, thus freeing up computational resources for the rest of the system and making the system deployable even in embedded contexts with limited resources.

PoseNet [76] is a collection of quantized pose estimation models based either on the MobileNetV2 or ResNet50 architectures, using the COCO encoding for the estimated keypoints locations. Output is in particular represented in the space of the resized input

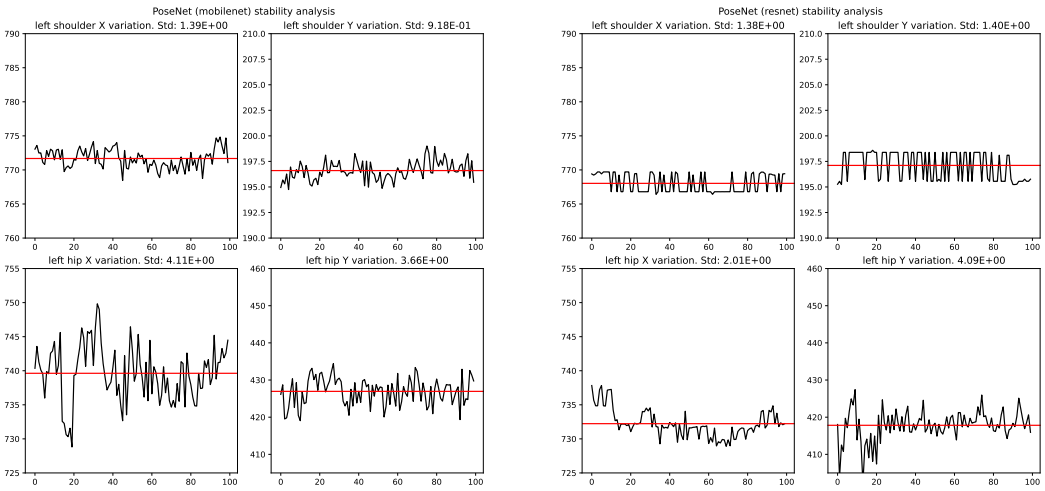
Figure 5.17: Front and back view of the Coral USB accelerator, on which the PoseNet library has been tested

image, and a rescaling operation is hence necessary to retrieve the non-distorted skeleton. A confidence score ranging between 0.0 and 1.0 is also associated to each keypoint and can be used to understand to which extent its position is accurate and if it is actually visible.

As described in the paragraph above, models available in the Coral PoseNet are quantized. This term refers to the technique of *quantization*, aimed at shrinking the model's size and latency while preserving its accuracy, by reducing the precision of numbers used in the computation. This process is also necessary to make the model deployable on the chip of the external Coral accelerator.

Both flavours have been tested, collecting the estimated poses and inference latencies on the aforementioned reference video. Results of the stability analysis are shown in figures 5.18. It's worth mentioning, however, that the model based on ResNet50, even if providing a more stable output, is characterised by a higher latency, in the order of seconds to estimate the pose on a single image. It is therefore not feasible to be implemented as a pose estimator due to the real-time requirements of the system.

Still regarding the inference latency, it is important the underline that the presence of an external hardware accelerator does not bring considerable improvements to the overall inference speed when compared to models deployed to the main computational unit. There is instead the risk that data transfer to and from the USB accelerator may become the bottleneck of the pose estimation module, requiring more time to copy data between the two memories than to actually process them.



(a) MobileNetV2 (b) ResNet50

Figure 5.18: Stability analysis of the x and y coordinates of the left shoulder and left hip landmarks estimated using the two models available in Posenet on a still subject

5.4.6. MediaPipe

MediaPipe is a set of open-source and cross platform ML solutions developed by Google, specifically targeted to real-time and live media use cases and optimized for deployment on mobile devices with constrained computational resources. As the name suggests, it is built around the concept of a pipeline and the framework allows the definition of a graph of modular components such as inference modules and media processing functions, but in the current context these multi-modal processing capabilities have not been fully exploited and only the pose estimation model has been used.

The model is based on the BlazePose project, which primarily addresses the common challenges of pose estimation in dancing or fitness contexts [11], such as the wide variety of possible poses and the numerous degrees of freedom and possible occlusions. In addition to the network itself, the project also introduces a new human body topology, ditching the 17 standard COCO keypoints listed in table 5.3 in favour of 33 landmarks, which are listed in figure 5.19 and are represented by their x and y coordinates together with a visibility confidence value *(ranging between 0 and 1)*.

From an architectural point of view, BlazePose is based on a detector-tracker pipeline: using a detector the model first locates the pose region-of-interest, subsequently predicting the aforementioned 33 keypoints. When loading the library it is possible to specify whether the model will be used on static images or videos; in the latter case the detection is performed only on the first frame or when body presence is lost, whereas subsequent

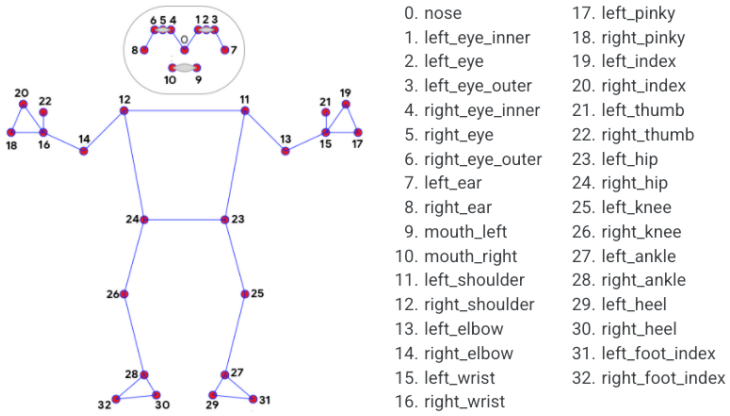


Figure 5.19: List of landmarks detected by MediaPipe and corresponding encoding

ones are derived from previously estimated keypoints. During usage it is also possible to specify the model's complexity, choosing among 3 flavours of BlazePose which offer different latency/accuracy tradeoffs.

Evaluation of the performance on the target deployment machine (that will be described in section 4.1) shows that the *Heavy* model does not bring any significant accuracy and smoothness improvement over the *Full* one, worsening instead the frame-rate. At the same time, the *Lite* version is not sufficiently good, since it occasionally loses track of some keypoints in fast or sudden movements.

Figure 5.20 shows the stability results of the model. Differently from the other tested

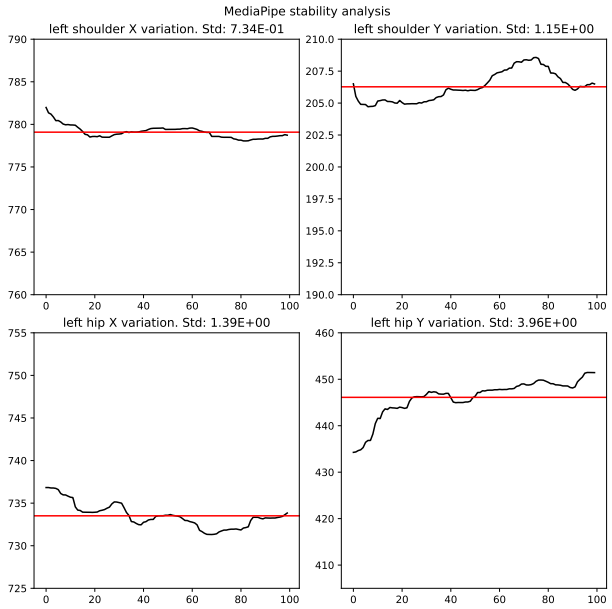


Figure 5.20: Stability analysis of the x and y coordinates of the left shoulder and left hip landmarks estimated using MediaPipe on a still subject

models, these results show a low jittering of the estimated landmarks' locations, producing a smooth estimation thanks to the built-in tracking mechanism.

5.4.7. Model selection

As already mentioned at the beginning of the section, model selection has been mostly based on the predictions' stability over time, to avoid noise in computed data due to jittering of estimated landmarks locations.

To estimate smoothness, besides a visual analysis of the plots reported in previous sections, the difference of landmarks' locations (δx and δy) in consecutive time steps has been computed. The standard deviation of these measurements has then been considered as a measure of jittering, showing the change of joints' positions over time on a still subject.

Model	Shoulder δx std	Shoulder δy std	Hip δx std	Hip δy std
Detectron2	1.1545	2.2620	1.1215	3.9311
OpenPifPaf	0.8748	0.9883	2.6859	5.2633
MediaPipe	**0.1316**	**0.1788**	**0.2266**	**0.4529**
PoseNet-MobileNetV2	0.7536	0.6141	2.0160	2.6253
PoseNet-ResNetV2	0.8810	1.0463	0.7299	2.7536

Table 5.5: Table summarizing the standard deviation of the difference of estimated x and y coordinates between two consecutive time steps of the left shoulder and hip, computed from the stability analysis on the video of a still subject for all the candidate models

Results are reported in table 5.5 and clearly show how MediaPipe outperforms other models in terms of stability and smoothness of estimated landmarks' locations. From the point of view of the inference latency, all the models have similar results, especially because several trade-offs are offered. For these reasons MediaPipe, being the one providing the smoothest predictions, has been chosen as the implemented pose estimator.

5.4.8. ROS integration

The selected pose estimation model has then been wrapped inside a ROS node, to make results available to other nodes on the *actor_pose* topic.

Since the model relies on images acquired through the webcam, it has been integrated into the vision macro-node as described in 5.2.

Output format

Output of the model has been reduced from the original 33 estimated keypoints to retain only those that have been identified as relevant to train the body-emotion recognition network, using the encoding shown in table 5.6.

Index	Landmark
0	Nose
1	Left shoulder
2	Right shoulder
3	Left elbow
4	Right elbow
5	Left wrist
6	Right wrist
7	Left hip
8	Right hip
9	Left knee
10	Right knee
11	Left ankle
12	Right ankle

Table 5.6: Table showing the list and encoding of the reduced set of body landmarks composing the actor's pose, estimated by the implemented pose estimation module

The estimated pose is represented as a 13×2 matrix, having the landmark indexes as rows and their x, y coordinates as columns, using the same input shape of the implemented neural network for body emotion recognition. However, to publish this data structure as a ROS message, a workaround is needed, using the *ros_ numpy* library [2] to convert numpy arrays to ROS-supported data types. In this particular case of a 2D array, the pose is exchanged as a Sensor_msgs/Image message, and both encoding/decoding is done in a single line of code, as shown in listing 5.3

Listing 5.3: Python code showing the usage of the ROS-numpy library to publish Numpy arrays as built-in ROS-messages

```
import ros2_numpy as rnp
from sensor_msgs.msg import Image

# Encoding of pose to publish
pose_msg = rnp.msgify(Image, pose, encoding="64FC1")
# Decoding of received pose
decoded_pose = rnp.numpify(pose_msg)
```

Pose normalization and landmarks location

In order to make the estimated pose invariant to the distance of the actor from the robot (*i.e. a closer actor will have a higher distance between landmarks*), normalization is performed and coordinates of landmarks are mapped from the image plane to a reference system centered in the hips, as shown in figure 5.21. This representation is also motivated by the representation of samples in the dataset on which the body emotion classifier has been trained, that uses as well an hip-centered coordinate system.

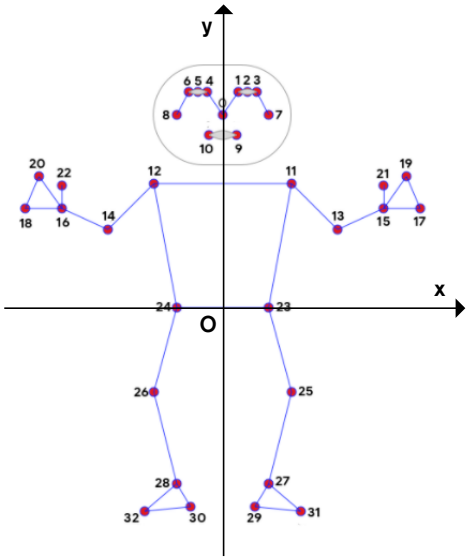


Figure 5.21: Diagram showing the coordinate reference system placed in the center of the hips used to represent location of landmarks in the output of the pose estimation module

To make the body emotion classifier behave consistently, the same normalization procedure has been applied both in training and inference. The method is in particular based on the one provided in the official MediaPipe documentation, aimed at making normalized poses have the same torso size and orientation, which has been modified to be compatible with the reduced set of joints listed in table 5.6.

5.5. Eye contact detection

To implement eye contact detection two main approaches have been followed, a geometric one, based on the position of the actor's facial landmarks in the field of view of the robot, and another one based on a deep neural network (DNN) trained on gaze datasets.

5.5.1. Geometric approach

This approach is based on the consideration that when people are looking directly into the camera their pupils are usually at the same distance from the vertical axis of the face (approximable to the distance between eyeballs and nose). In other terms, this implies that the position of the eyeballs is roughly *symmetrical* in the presence of eye contact, shifting instead to one side or the other if the person is looking left or right; figure 5.22 shows this behaviour for different orientations of the head.

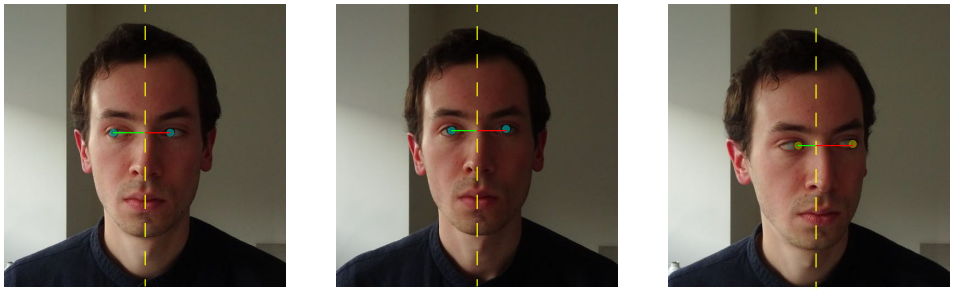


Figure 5.22: Sequence of images showing how symmetry of eyeballs with respect to nose axis changes according to different directions of gaze (namely to the left, to the front and to the right)

In particular, a dissymmetry index has been defined, ranging between 0 and 1 and computed using the formula in 5.10. The formula uses the 2 distances between the detected eyeballs and the nose: $d(p_{lefteye}, p_{nose}) = d_{left,nose}$ and $d(p_{righteye}, p_{nose}) = d_{right,nose}$, and computes the index by considering how different they are from each other. The result is in fact 0.5 when the eyes are symmetrical and becomes closer to 0 or 1 when the gaze is namely to the right or to the left.

$$Dys = \frac{d_{left,nose}}{d_{left,nose} + d_{right,nose}} \tag{5.10}$$

Using this information, different regions of this 0-1 interval can be defined, to map the index into a discrete classification of the gaze orientation. Facial landmarks used to compute the index are, moreover, already computed by the chosen pose estimation model described in 5.4 and hence no additional data source or extraction step is required. The model, in particular, provides the landmarks of the two eyeballs and the nose, which can therefore fed into the formula in 5.10 without any further manipulation.

Issues

This approach, however, suffers several issues due to the planar geometry used by the 2D pose estimation model, which makes impossible to distinguish whether the person is looking straight, up or down. As it can in fact be seen in figure 5.23, eyeballs in the exact same position can have different gaze orientations; they all have a dissymmetry index of around 0.5, but only in one case the person is keeping eye contact with the camera.

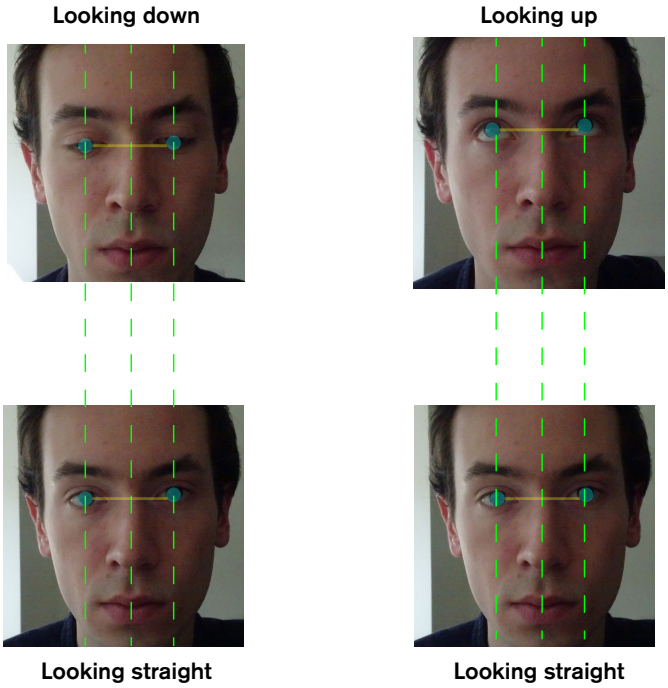


Figure 5.23: Comparison of different vertical gaze orientations sharing the same symmetry even when eye contact between the person and the camera is not present

Another problem, related to the chosen pose estimation model, is the fact that the profile view of a person yields the same dissymmetry as a front-facing face, due to the fact that the coordinates of the hidden eye are inferred by the model and placed really close to those of the visible landmark. This behaviour, given the fact that no information is provided by the model about the visibility of landmarks, makes it difficult to distinguish between the two cases and raises the need of a *face detector* to help disambiguate. This introduction, however, would add latency to the whole process. Since the face detection step is the most computationally expensive of most gaze estimation architectures and given the fact it would have anyway to be added, image-based techniques have been explored, to compensate the additional computational cost with a more robust and reliable estimation.

5.5.2. DNN approach

To overcome these limitations of the geometry-based approach, a deep neural network (DNN) has been used, namely the one introduced in [24]. The network is aimed at *"automatically detecting eye contact in egocentric video"* and is claimed to be the first to have reached an accuracy equivalent to that of human experts, scoring an overall precision of 0.936 and a recall of 0.943.

For this purpose a dataset has been built between 2015 and 2018 in the Georgia Tech Child Study Lab in Atlanta, GA and the Center for Autism and the Developing Brain in White Plains, NY, featuring 10 subjects both neurotypycal and affected by Autism Spectrum Disorder (ASD). Data, in particular, have been collected for each subject during two play interactions with a trained examiner wearing a pair of camera glasses (with no need of camera calibration). Frames from all the sessions have then been labelled by human coders: for the training set, the label has been assigned by a single human rater, whereas for the validation set it is based on the majority vote of multiple human raters.

The implemented convolutional neural network is instead based on a ResNet-50 [52] backbone architecture, accepting in input cropped face regions resized to 224x224 pixels. One of the peculiarities of the network and its training process is the fact that it has been carried out in two stages, to support the hypothesis that the model could leverage the high variability of face detection datasets to learn to model the relationship between head pose and gaze direction. For this reason, prior to fine-tuning the network on the dataset described above, the network has been trained on the publicly available MPIIFaceGaze [119] and EYEDIAP [42] datasets to regress the 3D gaze direction and on the SynHead [47] dataset to regress the 3D head pose. The project's code is publicly available on the Github platform and the network is implemented using the PyTorch framework.

To extract from each frame the face patches used as input by the network, the original implementation relies on the Dlib package, using its CNN-based face detector. This model, however, as reported by data collected on the development machine, introduces a considerable latency in the handling of every single frame, hindering the real-time capabilities of the whole eye-contact-detection pipeline. However, as explained in 5.2, face patches are already provided by the face detector (namely OpenCV) implemented in the vision node, which noticeably reduces the overall latency, making the pipeline work in real-time.

Eventually, the output of the network, differently from the previous approach, is not anymore a discrete classification of the gaze orientation, but a continuous value from 0 to 1 which tells the probability of eye contact presence in the given frame.

5.5.3. ROS integration

Once the eye-contact detection model has been defined, it has been integrated inside the ROS system, to make its results available to other consumer nodes. Since the chosen approach is based on the use of images collected through the webcam, the model has been wrapped inside the *vision* macro-node described in Section 5.2, running in a separate thread. Because the model yields a continuous probability of eye contact presence, the node's output is published as a decimal number (float64) on the *eye_ contact* topic.

5.6. Proxemics

Even though data and algorithms used to characterise the proxemics of the actor-robot interaction on the stage have been extensively described in 2.3, this section explains how they have been implemented in the overall architecture, showing from which hardware sensors the data are retrieved and which third-party components have been integrated.

The whole proxemics pipeline is based on the knowledge of the actor's position in the stage, expressed as relative to the robo-centric reference system. The first step in the implementation has therefore been to understand how this information could be retrieved using the laser sensors mounted on the robot.

Laser sensors are in fact often used to perform obstacle detection, namely detecting objects around the robot that are not part of the pre-loaded map of the environment, that should hence be avoided. The same approach can thus be reused, treating the presence of the actor as an "anomaly". Moreover, the context of the work and the fact that the robot operates on a controlled stage with a single human actor allows the definition of some preliminary simplifying assumptions:

- the actor's lower body part is an obstacle in the environment
- no other obstacle is present, due to the lack of other actors or moving elements
- the position and proxemics of the actor's lower body parts is a good approximation of their overall proxemics

These assumptions are important also in light of how lasers are mounted on the robot. They are in fact fixed directly on top of the base platform hosting the motors and the power supply, at a height corresponding to the shin of an adult person, and they can therefore detect only the presence of legs in the surrounding space.

Moreover, as previously explained, the robot's platform and architecture have been reused

from the RoboTower project [14], where it was crucial to know the exact location of the user, both for safety and gameplay reasons. To achieve this goal the control software of RoboTower, developed using ROS1 as well, features a player tracking mechanism based on the ROS *leg_detector* package, which uses a machine learning approach to identify leg-like patterns of laser scanner readings.

A similar approach has therefore been reused in this work, implementing a pre-trained leg detector to estimate the actor's position from the incoming stream of laser scans.

5.6.1. ROS leg detector

Since the release of the RoboTower project, the original leg detector package has been improved by *Leigh et al.*, who proposed a new approach based on tracking both legs [65], which is claimed to improve reliability (especially in cases of self-occlusion) and does not require a person to continually move in order to be tracked, which is of great importance in the context of theatre, where the interaction can often feature the two performers standing still one in front of the other.

This algorithm has been made publicly available on the Github platform and distributed as a ROS1 package and, as before, the package connects to the laser scans topic and publishes the estimated pose of the surrounding people.

Leg detector deployment and integration

The tracker, in particular, is deployed as a stand-alone node which receives data from the merged laser scans on the *scan* topic, and outputs detected people on the *people_tracked* topic, as an array of objects whose structure is shown in code 5.4.

Listing 5.4: Format of messages describing the estimated pose of a person detected by the implemented leg tracker package

```
geometry_msgs/Pose pose
uint32 id
```

As it can be seen in the code, each person is described by an identifier, to disambiguate between tracked individuals, and a *Pose* object, containing information about the spatial coordinates (Point) and orientation (Quaternion).

A final remark, however, has to be made about the tracker's deployment. As previously said, the package targets version 1 of the ROS framework, and is hence not directly compatible and integrated with the remainder of the system, targeting instead version 2.

To close this gap is therefore necessary the use of the bridge described in 4.4 and two architectural choices are possible about its placement:

1. Using the bridge to map laser scans from ROS1 to ROS2, keeping only the laser data acquisition and merging inside the ROS1 context and moving people tracking to ROS2
2. Using the bridge to map tracking results, keeping the leg tracker in its original ROS1 implementation

In order to implement the first option, a ROS2 porting of the leg tracker packages has been tested, but due to issues of dependencies still missing in ROS2 for some low-level features, the final choice has been to retain the original leg tracker implementation and to use the bridge to map its results.

However, messages through which detection results are provided are not native ROS data types, but custom objects defined inside the package. For this reason, it is not possible to directly install and use the bridge from the official ROS repositories, but it has to be compiled from source, loading the custom-defined message interfaces to be mapped.

Figure 5.24 summarizes how the two environments have been linked through the bridge.

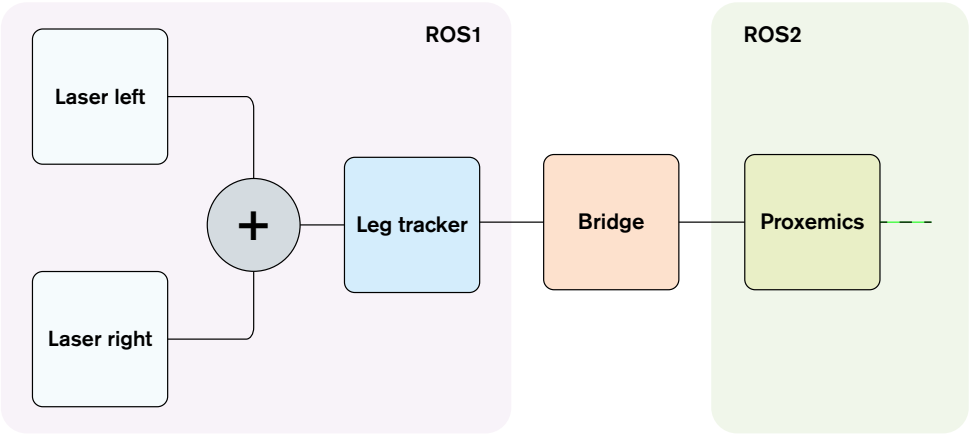


Figure 5.24: Diagram showing how the leg tracker and the bridge between ROS1 and ROS2 have been deployed in the final system architecture

5.6.2. Localization of robot in the stage

As described in 4.5.3, localization of the robot is provided by a third-party SLAM (*Simultaneous Localization And Mapping*) module, publishing the estimated coordinate frames on the *tf* topics. The position of the robot in the stage (namely its pose in the map

coordinate systems) is in particular computed by combining the *base_ link* (located in the center of the robot) and *map* (located in the origin of the map, at the lower-left corner of the stage) coordinate frames, as shown in figure 5.25. Computation is realized through the *tf2* ROS package, which offers a method (named *lookup_ transform*) to easily compute the translation between two given coordinate frames.

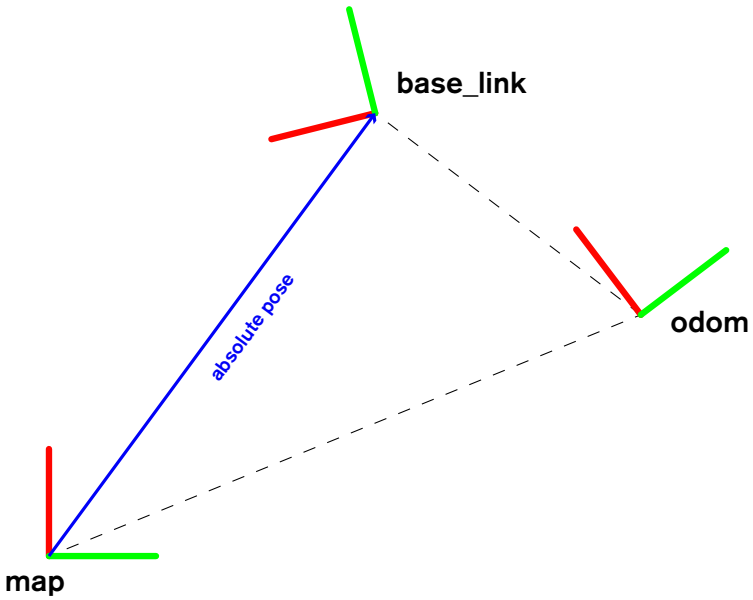


Figure 5.25: Diagram showing the coordinate frames published by the third-party SLAM module on the *tf* topic and the computed absolute pose in the map

5.6.3. Computation of proxemics descriptors

Once the deployment has been defined, and information about the actor's location has been successfully added as an input to the system, a specific node for the computation of proxemics descriptors has been developed.

The node, in particular, taking as input the x and y coordinates of the actor (if detected by the robot) in the robo-centric reference system and the pose of the robot in the map coordinate system, outputs the following information, each published in its own topic and with its own encoding:

- **Proxemic zone** occupied by the actor, published on the *proxemic_ zone* topic and represented as an unsigned integer with 3 coded values:
  - 0: for the personal and intimate zone
  - 1: for the neutral zone

  - 2: for when no interaction is present

- **Speed and direction** of the movement along the axis connecting the actor and the robot, published on the *proxemic_movement* topic and represented as a signed floating point number, where negative values correspond to an approach and positive values to a retreat
- **Discrete position** of the actor around the robot (as described in 2.3), published on the *actor_position* topic and represented also in this case by an unsigned integer with 4 coded values:
  - 0: for the frontal region
  - 1: for the left region
  - 2: for the back region
  - 3: for the right region
- **Raw x and y coordinates** of the actor's position in the robo-centric reference system, to be used by the actuator system in planning a physical reply
- **Position on the stage** occupied by the robot, following the labeling shown in figure 2.15 and encoded with an unsigned integer ranging from 0 to 9.

Incoming data are collected using the windowing mechanism described at the beginning of the chapter, but the node has been designed following a strategy design pattern to allow the future definition of different computation algorithms, based on other segmentation techniques rather than sliding windows. In the current implementation, data are processed as soon as the window is filled, using the algorithms described in 2.3 to compute the descriptors listed above.

5.7. Scenic action classification

In light of the considerations made in 3.5, the action classification has been implemented as a fuzzy-logic inference system, receiving in input all the relevant features and outputting the action vocabulary index and label of the classified scenic action.

5.7.1. Fuzzy logic

Before describing how action classification has been implemented and subsequently integrated in the ROS system, a brief introduction to fuzzy logic has to be made.

Background

Fuzzy logic (FL) is based on the fuzzy sets theory, introduced by Zadeh in 1965 [117] to model approximate concepts, that conceived sets to which elements can belong not in black-white / in-out fashion, but at different degrees of membership. Differently from crisp sets, having a binary membership function (MF) based on the properties of each element, fuzzy sets have a MF μ ranging in the $[0, 1]$ interval and computed as a function of the element's attributes (*e.g.* $\mu_{set}(X) = f(Attr(X))$). A MF defines a single fuzzy set, which is identified by a *linguistic label*.

FL extends this concept to logical propositions, whose variables do not have a binary truth value, but instead can have different degrees of truth represented by real numbers in the $[0, 1]$ interval. By doing so, FL is an example of an *infinite-valued* logic (beloging to the larger family of many-valued logic), featuring a continuum of truth values instead of the Boolean truth and false labels of the propositional or first/second-order logics.

Structure

FL is often used to model vagueness, imprecision and uncertainty in automatic models, mimicking the way people make decisions based on imprecise and non-quantitative information. This parallelism with human reasoning schemes is also evident in the structure of its propositions, which are of the form A **is** L, where A is a *linguistic variable* and L is a label associated to a fuzzy set.

A linguistic variable is in particular a tuple of 5 elements $(X, T(X), U, G, M)$ where:

- X is the name of the variable
- $T(X)$ is the set of terms for X, each corresponding to a fuzzy set
- U is the *universe of discourse*: all the possible values the base variable u can assume
- G is the syntactic rule to map each value of u to its interpretation
- M is the semantic rule to associate X to its meaning

As already mentioned, variables can then be used in propositions, whose truth value is not a crisp boolean value but instead a fuzzy set defined on $[0 \dots 1]$. Prepositions can then be combined with *modifiers*, which can increase or reduce the truth value of a given proposition by applying a continuous function on the mentioned truth value.

All these elements are the basic blocks with which fuzzy rules, and subsequently fuzzy inference systems, are built. An inference rule is, at high level, a mapping from an input

to an output, having the shape of *IF <antecedent> THEN <consequent>*, where each term is a set of logical clauses (propositions) related by logical operators such as AND, OR, NOT. In the context of fuzzy logic, clauses are called *linguistic clauses* and have the form of *V is L*, where *V* is a linguistic variable as described above and *L* (named *label*), is a possible fuzzy set value of *V*.

A fuzzy inference system is then composed of several rules, to which different weights can be given. These rules, according to the actual value of variables in the antecedents, produce different degrees of membership for possible values of the output (usually represented as singleton sets to make the computation faster) and hence after aggregation of these degrees an additional step is needed to retrieve a single specific value. This process takes the name of *defuzzyfication* and can be implemented with several operators, such as centroid, average of maxima, center of highest area, etc...

5.7.2. ROS integration

The action classifier module represents the system's interface with third-party actuator systems, responsible of exploiting the classified scenic action to generate a reply and make the improv scene proceed. Because of its nature, it gathers partial results and features computed by the other nodes to publish on a single topic the information about the classified scenic action.

Since, as already explained, not all the computed features and descriptors come into play in defining a scenic action, the node is subscribed only to the following topics:

- *emotion*
- *proxemic_zone*
- *proxemic_movement*

from which messages are read and used to update an internal copy of the corresponding features. Classification results are instead published on the *scenic_action* topic and are represented by the message type shown in listing 5.5, which contains the scenic action represented as both its vocabulary index (an unsigned integer) and a human-readable label (a string) for debugging purposes and more immediate feedback during evaluation.

Listing 5.5: Format of messages describing the classified scenic action published as the output of the system

```
uint8 action
std_msgs/String label
```

Lifecycle management

In order to make the node react to incoming features and provide a classification on a regular basis, the inference system is wrapped inside an internal thread that spins periodically every second. At each turn, the local copies of features are given as inputs to the fuzzy inference system and the result is published as an output.

5.7.3. Fuzzy variables

Using the concepts described in the previous section the classification mechanism outlined in chapter 3 has been implemented, mapping the identified relevant features to linguistic variables and representing feature-action relationships as fuzzy rules.

The fuzzy logic engine powering the system's reasoning capabilities has been implemented through the *scikit-fuzzy* library, a collection of algorithms and APIs written in the Python language and belonging to the Scikit stack.

As already mentioned, all the relevant features for action classification have been mapped to fuzzy variables, translating their discrete characterizations provided in 3.3 into fuzzy sets. For each variable the universe of discourse has been defined considering the corresponding feature's range of values (that could make sense in the context of an improv scene and inside the space of the particular stage used in this work), eventually reaching the definitions that are explained in the following sections.

Actor emotion

The estimated actor's emotion is represented as a vector of confidence scores, whose elements take value in the $[0 \dots 1]$ interval, and two approaches have been considered for its representation as a fuzzy variable.

Approach 1 - singletons The first approach has been to treat each emotion as a fuzzy set, defining singleton membership functions (namely very narrow triangular-shaped functions that can be approximated to a spike centred in the given value) each corresponding to one of the 7 emotions detected by the system. In this context, each emotion is linked to its index in the confidence vector, and only the dominant one is considered *(i.e. the one with the highest score, selected through an argmax operation)*, leading to the plot shown in figure 5.26.

This approach, however, does not take into account the confidence values assigned during inference to each emotion which, nevertheless, is already a measure of uncertainty that

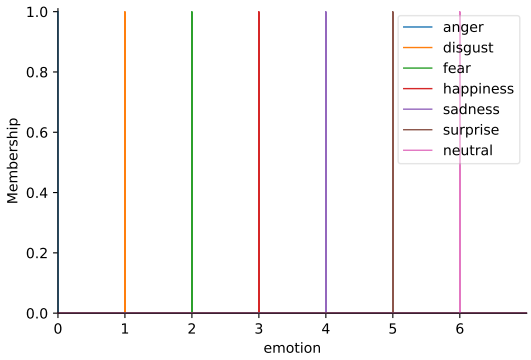


Figure 5.26: Plot showing the universe of discourse and singleton membership function for the emotion fuzzy variable

could be integrated into fuzzy reasoning.

For this reason, a second approach has been conceived, that could retain this information in the variable's formulation to make subsequent rules handle the emotion's uncertainty in a more accurate, meaningful and robust way.

Approach 2 - linear maps This second approach is in fact aimed at preserving the confidence value assigned to each emotion, using it as the corresponding truth value. To achieve this goal the previous formulation, based on a single universe of discourse on which different sets are defined for each emotion, is not expressive enough, and for this reason, 7 different fuzzy variables have been defined, each representing a single emotion class.

Each emotion variable is defined on a universe ranging from 0 to 1, having its assigned confidence score as the underlying base variable and a diagonal MF labeled "TRUE", as shown in figure 5.27.

Using this approach, it is possible to take into account the emotion detectors' uncertainty when computing the scenic action, enabling more complexity and expressivity if compared to the case when only the dominant one is used *(e.g. when "latent" emotions may contribute to the activation of more meaningful rules)*.

Proxemic zone

Proxemic zone is instead a categorical feature, and no uncertainty has been attached to it. This has been done because it is already the product of a smoothing mechanism (as described in 2.3) and hence no uncertainty can be derived for each characterization.

For this reason the singleton approach has been adopted for its representation, defining

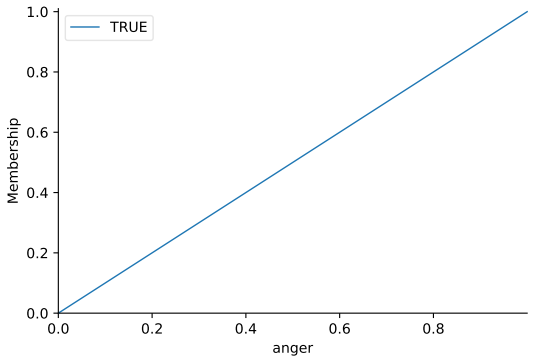


Figure 5.27: Plot showing the universe of discourse and membership function for a single-emotion fuzzy variable implemented as a linear map

three MFs corresponding to the three proxemic zones, as shown in the plot in figure 5.28.

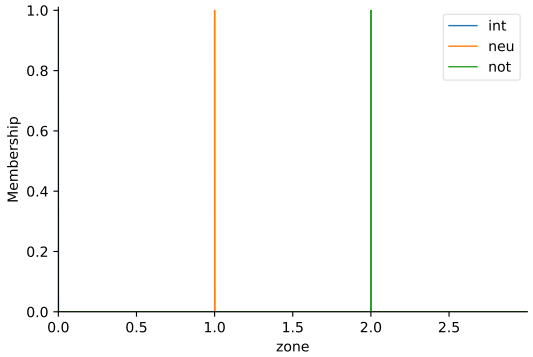


Figure 5.28: Plot showing the universe of discourse and membership functions for the proxemic zone fuzzy variable

Actor movement

Actor movement is a fuzzy variable that combines information about both the speed and direction of the actor moving on the segment connecting them with the robot. The base variable is the speed along this segment, computed using the multi-derivative approach explained in 2.3.1 and its universe of discourse has been defined through an empirical calibration, by moving around the robot at different speeds and measuring the corresponding values. The universe has then been divided into 7 sets, corresponding to 3 different speeds in both the approach and retreat directions and an additional label representing the absence of movement, as shown in figure 5.29.

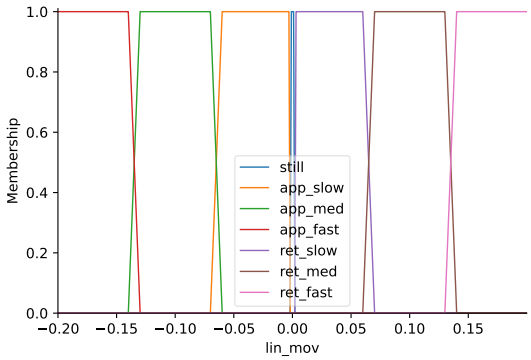


Figure 5.29: Plot showing the universe of discourse and membership functions for the actor's movement fuzzy variable

Output

The output of the system is the classified scenic action and, as done in many fuzzy classifiers, it is represented with the same singleton approach used for the proxemic zone: each action is encoded by its index in the action vocabulary, with MFs centered in that specific value, as shown in figure 5.30.

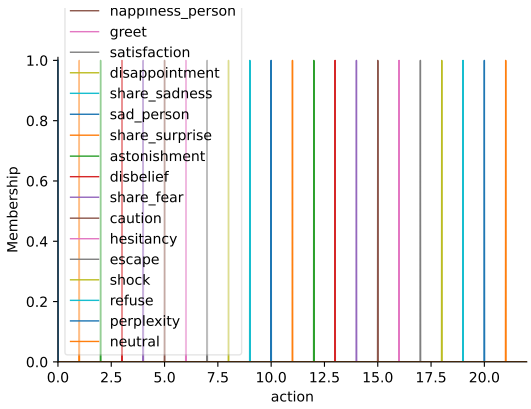


Figure 5.30: Plot showing the universe of discourse and membership functions for the scenic action output

5.7.4. Fuzzy rules

After having specified all the input and output variables as antecedents or consequents of the fuzzy inference system, rules to combine the two have been written by mapping table 3.1 into the fuzzy logic formalism. An example is given in listing 5.6, while the full list can be found in chapter A of the appendix.

Listing 5.6: Example fuzzy rules implemented in the inference system

```
ctrl.Rule(
    lin_mov["app_fast"] & zone["int"] & anger["TRUE"],
    action["attack"]
)
ctrl.Rule(
    (((lin_mov["app_slow"] | lin_mov["app_med"]) & zone["int"]) |
    (lin_mov["app_fast"] & zone["neu"] )) & anger["TRUE"],
    action["intimidation"]
)
```

The example also clearly shows how emotions are handled using the representation described above: instead of having a single variable named *"emotion"* for which is considered its membership to a certain class, single variables representing each emotion are used, whose confidence value is integrated into the numerical computation of the rule's activation.

Emotion smoothing

Regarding the emotion variables, an additional smoothing step has been implemented, to make the system more stable to noise in detecting emotional data. This approach, in particular, makes the system retain the previous emotion vector (hence ignoring a new detection) if the confidence about the old dominant emotion has not changed in a considerable way *(i.e. above a certain threshold)*. In mathematical terms, let i be the index of the previous dominant emotion and $E_t[i]$ and $E_{t-1}[i]$ the previous and current confidence values, the difference $\Delta_E(t) = E_t[i] - E_{t-1}[i]$ is computed and compared against a threshold K_E. If $\Delta_E(t) < K_E$ then the emotion vector (and hence the confidence values) used as input of the inference system is not changed, making it instead react to changes in the other variables.

Defuzzyfication

Since the goal of the implemented fuzzy system is to provide a classification of scenic actions in the ongoing interaction, which already correspond to the fuzzy sets associated with the output variable, defuzzyfication has been implemented in the form of a *linguistic approximation*. Using this approach the classified action is selected as the one corresponding to the fuzzy set closest to the defuzzyfied numerical value using, in particular, a rounding function to retrieve the indexing of the action in the vocabulary.

5.8. Final implemented architecture

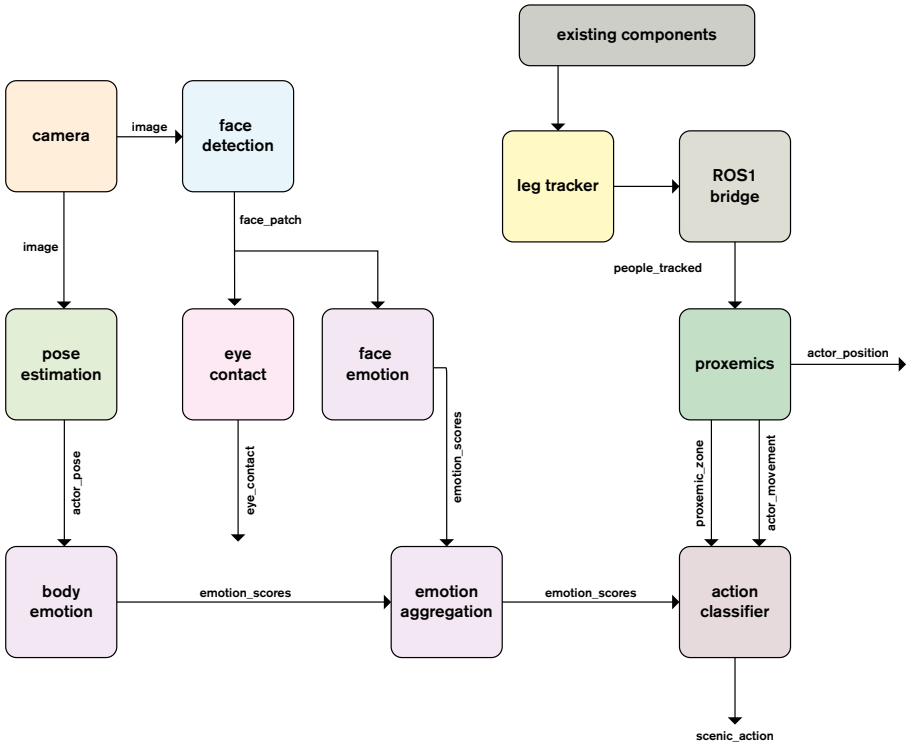


Figure 5.31: Diagram showing the final architecture implemented in the system, showing all the ROS nodes along with messages exchanged between them

Figure 5.31 shows a diagram of how all the ROS nodes described so far have been put together into the final architecture of the implemented system.

This architecture has been designed following the principles of modularity and separation concerns (where possible and not hindered by hardware or computational issues, as described in 5.2). Each input feature described in chapter 2 has in particular been linked to a specific ROS topic and computed or processed by a specific node. In this way it will be possible in the future to extend the system's capabilities with minimal effort, adding new input features and processing pipelines.

An example can be the addition of audio capabilities (realizing the auditory layer described in 1.1), to characterise for instance the emotional valence of the actor's speech. A node, connected to a microphone, can in fact be added to the system and compute the emotion associated with acoustic (*e.g. spectral analysis*) and semantic (*e.g. natural language processing*) features of the speech. This emotion can then be published on the *emotion_scores* topic and merged by the aggregator node (defining a proper fusion strategy) with the other emotions inferred from bodily and facial cues, to eventually classify

a global emotional state.

From the diagram (in which software already deployed on Robocchio and described in 4.5 has been represented with a single block) we can identify two main parts of the architecture:

- Emotion classification pipeline, classifying the emotion expressed by the actor from their pose and facial expressions acquired from a camera
- Proxemics description pipeline, computing quantitative and qualitative descriptors of the actor's spatial relationship with the robot and the stage, using data from laser sensors (pre-processed by a third-party module to extract people's locations)

These two parts eventually merge their results in the action classifier, which takes as input all the computed features to classify with a label the action currently happening in the ongoing interaction.

6 | Conclusions and future developments

And did you get what
you wanted from this life, even so?
I did.
And what did you want?
To call myself beloved, to feel myself
beloved on the earth.

Raymond Carver - Late Fragment

6.1. Conclusion

In this document, we have described the design and implementation of a system that makes an autonomous robot able to classify the scenic action performed by a human actor that is performing with it on a theatrical stage in the context of a (simplified) improv session.

The goal of this work was to build a system that could make an autonomous robotic actor aware of both its presence on the stage and everything happening around it. To achieve this goal the first step has been to build a conceptual framework, describing the relevant elements that come into play in understanding human actions and expressions, focusing in particular on the domains of acting and story-telling. From this framework, a physical system has been implemented (both from hardware and software points of view), able to map all the elements identified in the previous step into computational entities through proper sensors and data processing methods, to eventually provide a classification of the scenic actor performed by the human actor. This implementation, in particular, fully fulfils the initial requirements and goals, being able to classify the actor's actions using data retrieved from different sources *(e.g. laser sensors, facial expressions, body movements...)*, which have been identified from the study of literature in the fields of

theatre, psychology, choreography, and computer science.

In the end, the implemented system described in this document can be seen as the first building block in the development of a cognitive pipeline for an autonomous robot improviser. Prior to any discussion on the issues of personality, mood, and character of such a robot, is in fact necessary to lay a foundational architecture that could serve two distinct purposes:

- providing a stable ground to realize sensing capabilities, to make a robot recognize all the relevant elements of human communication and characterise the attitude and intentions of the actor through a set of archetypal actions
- validating the approach taken in designing the followed reference framework, showing the feasibility of such approach, especially when facing computational and real-time constraints

For this reason, it has been preferred to focus just on laying out the foundational architecture, implementing a functioning perception system to be later extended with more complex processing and classification methods. The classification in output from this system can be used by other modules in the aforementioned pipeline, treating it in a black-box fashion to simulate, at a higher level, cognitive functions related to sociability, morality, and aesthetics.

The implemented architecture can then find its place even in contexts outside the theatrical realm. As described in [68], *"both theatre and HRI aim to replicate some elements of humanity"*, sharing a structural similarity that can be exploited to move back and forth between the social and theatrical dimensions. Both the disciplines are involved in the simulation of human communication schemes and mechanisms and this work (and the acquired knowledge) can therefore be transferred to the more general domain of human-robot-interaction (and even human-machine-interaction). The proposed conceptual framework and the corresponding implemented architecture can be used as a reference to enrich the perceptive capabilities of autonomous interactive robots, improve their *situational awareness* of both the interaction venue and the user's state and eventually provide a more engaging, timely and effective experience.

6.2. Future directions

As already explained in the previous section, due to limited time available for the development of a master thesis the effort has been put mainly into the design and implementation of a foundational awareness system, to provide a stable starting ground for the future ad-

dition of more complex modules. During brainstorming sessions and study of literature several interesting ideas have emerged.

Even though they remained in the hypothetical space of possibilities they are still worth to mention, to serve as a compass showing the possible future directions of this work. These hints are related either to data acquisition, processing and output classification, and are briefly described in the following paragraphs.

Addition of new inputs and semantical layers As described in section 1.1, the system has been designed to support the future addition of new types of inputs, to make the classification of scenic actions rely on other cues and features rather than those actually implemented and described in this document.

A possible improvement is therefore to add more sensors and processing nodes to implement the remaining layers described in 1.1, related for instance to the verbal (*i.e. natural language processing*) and non-verbal (*i.e. voice acoustic analysis*) use of sound, to mimicry (*i.e. activity recognition*) and to linguistic communication.

Besides these new sensory capabilities, it would be interesting to explore whether and how the features already detected by the system, but not explicitly used, can be meaningfully assimilated in the classification framework. This is the case for eye contact presence and actor's position, which can ideally add nuance to the current set of situations, even as a *second-step* refinement of the current outputs.

Awareness of meta-features Besides the sheer awareness of what is happening on the stage and the partner's attitude, a critical element in improvisation is being able to evaluate and assess the scene as if seen by an external viewer. This is important for understanding whether the story being narrated is engaging for the audience and at which point of its development the two actors found themselves. The latter information can be represented through the concept of *dramatic arc* [41] (shown in figure 6.1), which defines five main parts composing the dynamics of a *well-narrated* story.

These features would be placed at a higher conceptual level, a sort of meta-description of the acting describing its quality, which can be used either *on-line* to guide the selection of replies by third-party systems or *off-line* to implement for instance reinforcement-learning (RL) models and train policies to select which actions fit best in a given frame of tension.

The dramatic arc in particular can be largely exploited in HRi, as it well approximates the usual dynamics of the interactions [68]. A robot should therefore be aware of the current point of tension to select actions that make it grow or decrease accordingly, in

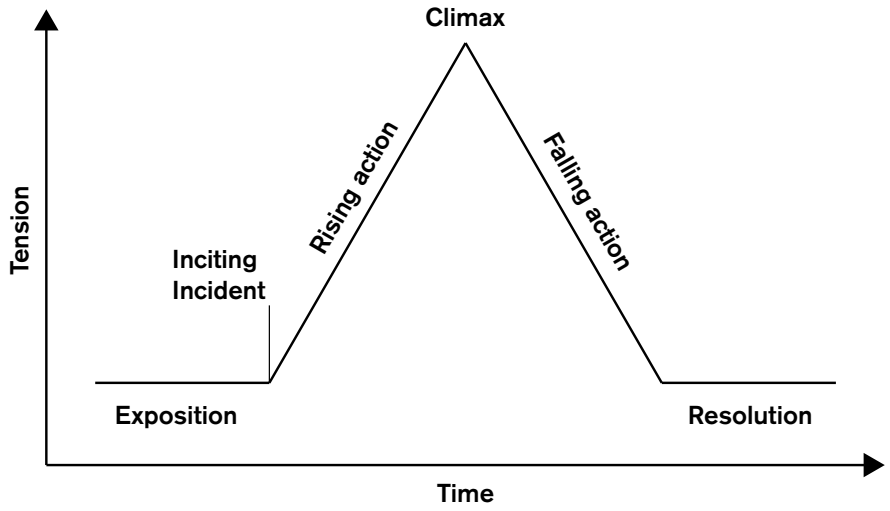


Figure 6.1: Dramatic arc identifying five main moments in the evolution of tension in a story, proposed by *Freytag* in [41]

order to create a more natural and friction-less experience.

Linked to this concept, another interesting metric that can be explored and added to this set of meta-features is the *variability* of a story over time. A common rule of improvisation is in fact to leave, in the first stages of the scene, a door open to any possibility of development, leaving space for the ambiguity of interpretation and avoiding to ascribe specific meaning to the peers' actions. The first part of a story is therefore characterised by a high variability: the audience becomes acquainted with characters and their mutual relationships start to develop or to become more detailed; in this stage, a robot improviser should select replies that do not narrow this field of possibilities and instead introduce new elements to the scene to be used later on. The second part, instead, is concerned about making all the introduced elements collapse to a single ending point, closing the issues left open and giving a final meaning to everything that has happened; actions chosen in this context should therefore aim to restrict the possibilities, making everything collapse to a smaller set of interpretations. A possible way to implement this awareness and describe this variability index is through the concept of *entropy*, which is usually employed in computer science and probability theory to describe the informational content of some element. Inspired by the work done in [32] a proposal for a future enhancement is, therefore, to make the system able to compute the entropy associated with each scene and the impact each action has on it, enabling other systems to plan actions in the space of entropy.

Ontology of archetypical scenes This concept of entropy can be further extended by making the system able to reason on the specific structure of the scene, representing it through the use of *descriptive* or *modal logics*. A possible approach might be to create an ontology describing at high-level the elements that can occur in a scene (*e.g. characters, relationships, events...*), even in a hierarchical way in a fashion borrowed from object-oriented-programming, to reason on this world models for assessing the inherent entropy of the scene and the available affordances for its development (*e.g. which actions are feasible, which other worlds are reachable from the current one...*).

Dataset of scenic action to train more complex models Regarding the output of the system, as already pointed out in section 3.5, the use of classifiers based on deep-learning methods would require a proper dataset, annotated with the supported scenic action sequences of observations. A work of this kind would have been too time-consuming to be carried out at an acceptable level in the context of this thesis but is nevertheless relevant and necessary for further improvements of the system's sensing capabilities. The dataset can in particular be developed following the approach of [118], employing a diverse group of actors performing scenic actions and collecting all the measurable and relevant variables described in the developed framework.

With this dataset, it would be possible to train models more complex than a fuzzy inference system, not only to realize a neural classification of the scenic actions but also to enable a neural synthesis of the robot's replies. Training in fact a generative adversarial network (GAN) it would be possible to make the robot autonomously come up with different ways of performing a given scenic action (*i.e. determining how to use proxemics, which dominant emotion to express, how to use sound, which movement to perform...*), removing the human element from the definition of replies and thus getting closer to the goal of a fully autonomous robot actor.

Bibliography

[1] ros2/ros1_bridge: Ros 2 package that provides bidirectional communication between ros 1 and ros 2, . URL `https://github.com/ros2/ros1_bridge`.

[2] Box-robotics/ros2_numpy: Tools for converting ros messages to and from numpy arrays, . URL `https://github.com/Box-Robotics/ros2_numpy`.

[3] I. Ajili, M. Mallem, and J.-Y. Didier. Human motions and emotions recognition inspired by lma qualities. *The Visual Computer*, 35(10):1411–1426, 2019.

[4] I. Alcubilla Troughton, K. Baraka, K. Hindriks, and M. Bleeker. Robotic improvisers: Rule-based improvisation and emergent behaviour in hri. In *Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction*, pages 561–569, 2022.

[5] and European Data Protection Supervisor, A. Horvath, and K. Vemou. *EDPS TechDispatch : facial emotion recognition. Issue 1, 2021*. Publications Office, 2021. doi: doi/10.2804/014217.

[6] O. Arriaga, M. Valdenegro-Toro, and P. Plöger. Real-time convolutional neural networks for emotion and gender classification, 2017.

[7] O. Arriaga, M. Valdenegro-Toro, M. Muthuraja, S. Devaramani, and F. Kirchner. Perception for autonomous systems (paz), 2020.

[8] A. P. Atkinson, W. H. Dittrich, A. J. Gemmell, and A. W. Young. Emotion perception from dynamic and static body expressions in point-light and full-light displays. *Perception*, 33(6):717–746, 2004.

[9] B. J. Baars. *A cognitive theory of consciousness*. Cambridge University Press, 1993.

[10] A. L. Ballardini, S. Fontana, A. Furlan, and D. G. Sorrenti. ira_laser_tools: a ros laserscan manipulation toolbox. *arXiv preprint arXiv:1411.1086*, 2014.

[11] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grund-

mann. Blazepose: On-device real-time body pose tracking. 2020. doi: 10.48550/ARXIV.2006.10204. URL `https://arxiv.org/abs/2006.10204`.

[12] A. Bechara. The role of emotion in decision-making: Evidence from neurological patients with orbitofrontal damage. *Brain and Cognition*, 55(1):30–40, 2004. ISSN 0278-2626. doi: https://doi.org/10.1016/j.bandc.2003.04.001. URL `https://www.sciencedirect.com/science/article/pii/S0278262603002859`. Development of Orbitofrontal Function.

[13] Z. Z. Bien and H.-E. Lee. Effective learning system techniques for human–robot interaction in service environment. *Knowledge-Based Systems*, 20(5):439–456, 2007.

[14] A. Bonarini, S. Boriero, and E. L. S. de Oliveira. Robot player adaptation to human opponents in physical, competitive robogames. pages 851–856. IEEE, 8 2020. ISBN 978-1-7281-6075-7. doi: 10.1109/RO-MAN47096.2020.9223576.

[15] L. Bonetti. Design and implementation of an actor robot for a theatrical play, 2021.

[16] E. Borgna. *L'arcipelago delle emozioni*, volume 314. Feltrinelli Editore, 2001.

[17] D. Bouchard and N. Badler. Semantic segmentation of motion capture using laban movement analysis. In *International Workshop on Intelligent Virtual Agents*, pages 37–44. Springer, 2007.

[18] C. Breazeal. Socially intelligent robots. *interactions*, 12(2):19–22, 2005.

[19] C. Breazeal, A. Brooks, J. Gray, M. Hancher, C. Kidd, J. McBean, D. Stiehl, and J. Strickon. Interactive robot theatre. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 4, pages 3648–3655. IEEE, 2003.

[20] A. Bruce, J. Knight, S. Listopad, B. Magerko, and I. R. Nourbakhsh. Robot improv: Using drama to create believable agents. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 4, pages 4002–4008. IEEE, 2000.

[21] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[22] M. Chen and J. A. Bargh. Consequences of automatic evaluation: Immediate behavioral predispositions to approach or avoid the stimulus. *Personality and social psychology bulletin*, 25(2):215–224, 1999.

[23] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[24] E. Chong, E. Clark-Whitney, A. Southerland, E. Stubbs, C. Miller, E. L. Ajodan, M. R. Silverman, C. Lord, A. Rozga, R. M. Jones, et al. Detection of eye contact with deep neural networks is as accurate as human experts. *Nature communications*, 11(1):1–10, 2020.

[25] A. S. Cowen and D. Keltner. Self-report captures 27 distinct categories of emotion bridged by continuous gradients. *Proceedings of the National Academy of Sciences*, 114(38):E7900–E7909, 2017.

[26] R. Cowie, E. Douglas-Cowie, N. Tsapatsoulis, G. Votsis, S. Kollias, W. Fellenz, and J. G. Taylor. Emotion recognition in human-computer interaction. *IEEE Signal processing magazine*, 18(1):32–80, 2001.

[27] M. Danesi. Proxemics. In K. Brown, editor, *Encyclopedia of Language & Linguistics (Second Edition)*, pages 241–243. Elsevier, Oxford, second edition edition, 2006. ISBN 978-0-08-044854-1. doi: https://doi.org/10.1016/B0-08-044854-2/01441-3. URL `https://www.sciencedirect.com/science/article/pii/B0080448542014413`.

[28] Q. Dang, J. Yin, B. Wang, and W. Zheng. Deep learning based 2d human pose estimation: A survey. *Tsinghua Science and Technology*, 24(6):663–676, 2019.

[29] C. Darwin and F. Darwin. *The expression of the emotions in man and animals*. Cambridge University Press, Cambridge, 2009. ISBN 9780511694110. doi: 10.1017/CBO9780511694110. URL `http://ebooks.cambridge.org/ref/id/CBO9780511694110`.

[30] B. De Gelder. Why bodies? twelve reasons for including bodily expressions in affective neuroscience. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1535):3475–3484, 2009.

[31] D. C. Dennett. *The intentional stance*. MIT press, 1989.

[32] M. Dinu. Entropy and prediction in the study of theatre. *Poetics*, 13(1-2):57–70, 1984.

[33] P. Ekman. An argument for basic emotions. *Cognition and Emotion*, 6:169–200, 5 1992. ISSN 0269-9931. doi: 10.1080/02699939208411068. URL `https://www.tandfonline.com/doi/full/10.1080/02699939208411068`.

[34] P. Ekman. Basic emotions. *Handbook of cognition and emotion*, 98(45-60):16, 1999.

[35] A. Fallatah, J. Urann, and H. Knight. The robot show must go on: Effective responses to robot failures. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 325–332. IEEE, 2019.

[36] S. Fang, C. Achard, and S. Dubuisson. Personality classification and behaviour interpretation: An approach based on feature categories. In *Proc. of ACM Int. Conf. on Multimodal Interaction*, pages 225–232, 2016.

[37] J. Ferreira, S. Brás, C. F. Silva, and S. C. Soares. An automatic classifier of emotions built from entropy of noise. *Psychophysiology*, 54(4):620–627, 2017.

[38] M. Fey and J. E. Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[39] P. P. Filntisis, N. Efthymiou, P. Koutras, G. Potamianos, and P. Maragos. Fusing body posture with facial expressions for joint recognition of affect in child–robot interaction. *IEEE Robotics and Automation Letters*, 4(4):4011–4018, 2019. doi: 10.1109/LRA.2019.2930434.

[40] I. I. for Restorative Practices. Nine affects. URL `https://www.iirp.edu/defining-restorative/nine-affects`.

[41] G. Freytag. Freytag's technique of the drama: An exposition of dramatic composition and art .(ej macewan, trans.). *Chicago: Scott, Foresman and Company, LCCN.(Original work published 1894)*, 1900.

[42] K. Funes Mora, F. Monay, and J.-M. Odobez. Eyediap: a database for the development and evaluation of gaze estimation algorithms from rgb and rgb-d cameras. pages 255–258, 03 2014. doi: 10.1145/2578153.2578190.

[43] M. J. Gielniak and A. L. Thomaz. Enhancing interaction through exaggerated motion synthesis. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 375–382, 2012.

[44] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[45] S. Goenaga, L. Navarro, C. G. Quintero M, M. Pardo, et al. Imitating human emotions with a nao robot as interviewer playing the role of vocational tutor. *Electronics*, 9(6):971, 2020.

[46] I. R. M. D. R. W. Group et al. Ieee standard for robot map data representations for navigation. 2015.

[47] J. Gu, X. Yang, S. De Mello, and J. Kautz. Dynamic facial analysis: From bayesian filtering to recurrent neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1548–1557, 2017.

[48] H. Gunes and M. Piccardi. Affect recognition from face and body: early fusion vs. late fusion. In *2005 IEEE international conference on systems, man and cybernetics*, volume 4, pages 3437–3443. IEEE, 2005.

[49] E. T. Hall and E. T. Hall. *The hidden dimension*, volume 609. Anchor, 1966.

[50] E. T. Hall, R. L. Birdwhistell, B. Bock, P. Bohannan, A. R. Diebold Jr, M. Durbin, M. S. Edmonson, J. Fischer, D. Hymes, S. T. Kimball, et al. Proxemics [and comments and replies]. *Current anthropology*, 9(2/3):83–108, 1968.

[51] N. Hatami, Y. Gavet, and J. Debayle. Classification of time-series images using deep convolutional neural networks. In *Tenth international conference on machine vision (ICMV 2017)*, volume 10696, page 106960Y. International Society for Optics and Photonics, 2018.

[52] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[53] C. Izard. Four systems for emotion activation: cognitive and noncognitive processes. *Psychological review*, 100 1:68–90, 1993.

[54] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678, 2014.

[55] K. Johnstone and I. Wardle. *Impro: Improvisation and the theatre*. Routledge, 2012.

[56] W. Ju, B. A. Lee, and S. R. Klemmer. Range: exploring implicit interaction through electronic whiteboard design. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pages 17–26, 2008.

[57] K. Katevas, P. G. Healey, and M. T. Harris. Robot stand-up: engineering a comic

performance. In *Proc. Workshop on humanoid robots and creativity, IEEE-RAS Int. Conf. on Humanoid Robots*. IEEE, 2014.

[58] H. Knight, S. Satkin, V. Ramakrishna, and S. Divvala. A savvy robot standup comic: Online learning through audience tracking. In *Workshop paper (TEI'10)*, 2011.

[59] S. Kreiss, L. Bertoni, and A. Alahi. OpenPifPaf: Composite Fields for Semantic Keypoint Detection and Spatio-Temporal Association. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–14, March 2021.

[60] R. Laban and L. Ullmann. The mastery of movement. 1971.

[61] D. Lambert. *Collins Gem - Body Language*. Collins, 2004. ISBN 0007189923.

[62] C. Larboulette and S. Gibet. A review of computable expressive descriptors of human motion. In *Proceedings of the 2nd International Workshop on Movement and Computing*, pages 21–28, 2015.

[63] R. S. Lazarus and B. N. Lazarus. *Passion and reason: Making sense of our emotions*. Oxford University Press, USA, 1994.

[64] J. LeDoux. *The emotional brain: The mysterious underpinnings of emotional life*. Simon and Schuster, 1998.

[65] A. Leigh, J. Pineau, N. Olmedo, and H. Zhang. Person tracking and following with 2d laser scanners. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 726–733. IEEE, 2015.

[66] J. S. Lerner, Y. Li, P. Valdesolo, and K. S. Kassam. Emotion and decision making. *Annual review of psychology*, 66:799–823, 2015.

[67] D. V. Lu. Ontology of robot theatre. In *Proc. Workshop Robotics and Performing Arts: Reciprocal Influences, ICRA*, 2012.

[68] D. V. Lu and W. D. Smart. Human-robot interactions as theatre. In *2011 Ro-Man*, pages 473–478. IEEE, 2011.

[69] L. J. Martin, B. Harrison, and M. O. Riedl. Improvisational computational storytelling in open worlds. In *International Conference on Interactive Digital Storytelling*, pages 73–84. Springer, 2016.

[70] J. Martinez, R. Hossain, J. Romero, and J. J. Little. A simple yet effective baseline for 3d human pose estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 2640–2649, 2017.

[71] C. Mikalauskas, T. Wun, K. Ta, J. Horacsek, and L. Oehlberg. Improvising with an audience-controlled robot performer. In *Proceedings of the 2018 designing interactive systems conference*, pages 657–666, 2018.

[72] N. Mitsunaga, C. Smith, T. Kanda, H. Ishiguro, and N. Hagita. Adapting robot behavior for human–robot interaction. *IEEE Transactions on Robotics*, 24(4):911–916, 2008.

[73] L. P. Naumann, S. Vazire, P. J. Rentfrow, and S. D. Gosling. Personality judgments based on physical appearance. *Personality and social psychology bulletin*, 35(12): 1661–1671, 2009.

[74] I. E. of Philosophy. Theories of emotion, . URL `https://iep.utm.edu/emotion/`.

[75] S. E. of Philosophy. Ancient, medieval and renaissance theories of the emotions, . URL `https://plato.stanford.edu/entries/emotions-17th18th/LD1Background.html`.

[76] G. Papandreou, T. Zhu, L.-C. Chen, S. Gidaris, J. Tompson, and K. Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the European conference on computer vision (ECCV)*, pages 269–286, 2018.

[77] H. W. Park, I. Grover, S. Spaulding, L. Gomez, and C. Breazeal. A model-free affective reinforcement learning approach to personalization of an autonomous social robot companion for early literacy education. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 687–694, 2019.

[78] R. Picard. *Affective Computing*. The MIT Press, 1997. URL `https://mitpress.mit.edu/books/affective-computing`.

[79] R. PLUTCHIK. Chapter 1 - a general psychoevolutionary theory of emotion. In R. Plutchik and H. Kellerman, editors, *Theories of Emotion*, pages 3–33. Academic Press, 1980. ISBN 978-0-12-558701-3. doi: https://doi.org/10.1016/B978-0-12-558701-3.50007-7.

[80] J. POSNER, J. A. RUSSELL, and B. S. PETERSON. The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology. *Development and Psychopathology*, 17(3):715–734, 2005. doi: 10.1017/S0954579405050340.

[81] B. Reeves and C. Nass. *The media equation: How people treat computers, television, and new media like real people*. Cambridge university press Cambridge, UK, 1996.

[82] J. Rett. *Robot-human interface using laban movement analysis inside a bayesian framework*. PhD thesis, Universidade de Coimbra (Portugal), 2009.

[83] P. Robbins. Modularity of mind. 2009.

[84] J. Rond, A. Sanchez, J. Berger, and H. Knight. Improv with robots: creativity, inspiration, co-performance. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1–8. IEEE, 2019.

[85] B. Rozemberczki, P. Scherer, Y. He, G. Panagopoulos, A. Riedel, M. Astefanoaei, O. Kiss, F. Beres, , G. Lopez, N. Collignon, and R. Sarkar. PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, page 4564–4573, 2021.

[86] J. A. Russell. A circumplex model of affect. *Journal of Personality and Social Psychology*, 39:1161–1178, 1980. ISSN 0022-3514. doi: 10.1037/h0077714.

[87] T. Salinsky and D. Frances-White. *The improv handbook: The ultimate guide to improvising in comedy, theatre, and beyond*. Bloomsbury Publishing, 2017.

[88] A. V. Samsonovich. Emotional biologically inspired cognitive architecture. *Biologically Inspired Cognitive Architectures*, 6:109–125, 2013.

[89] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. B. Wiltschko. A gentle introduction to graph neural networks. *Distill*, 6(9):e33, 2021.

[90] K. Scherer. Emotions, psychological structure of, 2001.

[91] K. R. Scherer. Appraisal theory. 1999.

[92] K. Schindler, L. Van Gool, and B. De Gelder. Recognizing emotions expressed by body pose: A biologically inspired neural model. *Neural networks*, 21(9):1238–1246, 2008.

[93] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*, pages 362–373. Springer, 2018.

[94] S. I. Serengil and A. Ozpinar. Lightface: A hybrid deep face recognition framework. In *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pages 1–5, 2020. doi: 10.1109/ASYU50717.2020.9259802.

[95] S. I. Serengil and A. Ozpinar. Hyperextended lightface: A facial attribute anal-

ysis framework. In *2021 International Conference on Engineering and Emerging Technologies (ICEET)*, pages 1–4, 2021. doi: 10.1109/ICEET53442.2021.9659697.

[96] A. Subasi. Eeg signal classification using wavelet feature extraction and a mixture of expert model. *Expert Systems with Applications*, 32(4):1084–1093, 2007.

[97] M. Suguitan and G. Hoffman. Blossom: A handcrafted open-source robot. *ACM Transactions on Human-Robot Interaction (THRI)*, 8(1):1–27, 2019.

[98] L. Takayama and C. Pantofaru. Influences on proxemic behaviors in human-robot interaction. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5495–5502. IEEE, 2009.

[99] M. Tkalčič. Emotions and personality in recommender systems: Tutorial. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 535–536, 2018.

[100] D. Tome, C. Russell, and L. Agapito. Lifting from the deep: Convolutional 3d pose estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2500–2509, 2017.

[101] S. Tomkins. *Affect imagery consciousness: Volume I: The positive affects*. Springer publishing company, 1962.

[102] S. S. Tompkins. Affect, imagery, consciousness: Ii. the negative affects. 1963.

[103] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014.

[104] A. Truong and T. Zaharia. Laban movement analysis and hidden markov models for dynamic 3d gesture recognition. *EURASIP Journal on Image and Video Processing*, 2017(1):1–16, 2017.

[105] A. Van Wynsberghe. Designing robots for care: Care centered value-sensitive design. *Science and engineering ethics*, 19(2):407–433, 2013.

[106] O. Vartanian, K. Stewart, D. R. Mandel, N. Pavlovic, L. McLellan, and P. J. Taylor. Personality assessment and behavioral prediction at first impression. *Personality and individual differences*, 52(3):250–254, 2012.

[107] J. Velsquez. Modeling emotions and other motivations in synthetic agents. *Aaai/iaai*, pages 10–15, 1997.

[108] J. Vilk and N. T. Fitter. Comedians in cafes getting data: evaluating timing and adaptivity in real-world robot comedy performance. In *Proc. ACM/IEEE Int. Conf. on Human-Robot Interaction*, pages 223–231, 2020.

[109] H. G. Wallbott. Bodily expression of emotion. *European journal of social psychology*, 28(6):879–896, 1998.

[110] S. Wang, J. Li, T. Cao, H. Wang, P. Tu, and Y. Li. Dance emotion recognition based on laban motion analysis using convolutional neural network and long short-term memory. *IEEE Access*, 8:124928–124938, 2020.

[111] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

[112] J. Willis and A. Todorov. First impressions: Making up your mind after a 100-ms exposure to a face. *Psychological science*, 17(7):592–598, 2006.

[113] L. Wu, S. L. Oviatt, and P. R. Cohen. Multimodal integration-a statistical view. *IEEE Transactions on Multimedia*, 1(4):334–341, 1999.

[114] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[115] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[116] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977.

[117] L. Zadeh. Zadeh, fuzzy sets. *Inform Control*, 8:338–353, 1965.

[118] M. Zhang, L. Yu, K. Zhang, B. Du, B. Zhan, S. Chen, X. Jiang, S. Guo, J. Zhao, Y. Wang, et al. Kinematic dataset of actors expressing emotions. *Scientific data*, 7 (1):1–8, 2020.

[119] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling. Mpiigaze: Real-world dataset and deep appearance-based gaze estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 41(1):162–175, 2019. doi: 10.1109/TPAMI.2017.2778103.

A | Implemented fuzzy rules

This chapter described the fuzzy rules implemented to classify the scenic actions performed by the actor, using the classical notation in the form `IF...THEN...`.

The following notation has in particular been used in the code:

- `lin_mov`: represents the movement variable, and possible values are represented by the terms `app_fast`, `app_med`, `app_slow`, `still`, `ret_slow`, `ret_med`, `ret_fast`, trivially mapping to stillness and different approach-retreat speeds
- `zone`: describes the proxemic zone variable, taking three possible values, namely `int` for the personal space, `neu` for the neutral one, and `not` in case of no interaction
- Emotions and actions are trivially represented by variables with their same name

```
IF (lin_mov[app_fast] AND (NOT-zone[not])) AND anger[TRUE] THEN action[
    attack]
IF (((lin_mov[app_slow] OR lin_mov[app_med]) OR lin_mov[still]) AND (zone[
    int] OR zone[neu])) AND anger[TRUE] THEN action[intimidation]
IF (lin_mov[still] AND (NOT-zone[int])) AND anger[TRUE] THEN action[
    scolding]
IF ((((lin_mov[ret_slow] OR lin_mov[ret_med]) OR lin_mov[ret_fast]) AND (
    zone[neu] OR zone[not])) OR zone[not]) AND anger[TRUE] THEN action[
    grudge]
IF happiness[TRUE] AND zone[int] THEN action[share_joy]
IF happiness[TRUE] AND zone[not] THEN action[happiness_person]
IF happiness[TRUE] AND zone[neu] THEN action[greet]
IF ((lin_mov[ret_slow] OR lin_mov[ret_med]) OR lin_mov[ret_fast]) AND
    happiness[TRUE] THEN action[satisfaction]
IF ((lin_mov[ret_slow] OR lin_mov[ret_med]) OR lin_mov[ret_fast]) AND
    sadness[TRUE] THEN action[disappointment]
IF zone[int] AND sadness[TRUE] THEN action[share_sadness]
IF ((NOT-zone[int]) AND (NOT-((lin_mov[ret_slow] OR lin_mov[ret_med]) OR
    lin_mov[ret_fast]))) AND sadness[TRUE] THEN action[sad_person]
IF (NOT-zone[not]) AND surprise[TRUE] THEN action[share_surprise]
IF (((lin_mov[still] OR lin_mov[app_med]) OR lin_mov[app_slow]) AND (NOT-
    zone[int])) AND surprise[TRUE] THEN action[astonishment]
```

```
IF ((lin_mov[ret_fast] OR lin_mov[ret_slow]) OR lin_mov[ret_med]) AND
    surprise[TRUE] THEN action[disbelief]
IF (lin_mov[app_fast] AND zone[not]) AND surprise[TRUE] THEN action[shock]
IF zone[int] AND fear[TRUE] THEN action[share_fear]
IF (((lin_mov[app_fast] OR lin_mov[app_med]) OR lin_mov[still]) AND (NOT-
    zone[not])) AND fear[TRUE] THEN action[share_fear]
IF (lin_mov[app_slow] AND (NOT-zone[not])) AND fear[TRUE] THEN action[
    caution]
IF (((lin_mov[still] OR lin_mov[app_slow]) OR lin_mov[ret_slow]) AND (NOT-
    zone[int])) AND fear[TRUE] THEN action[hesitancy]
IF (((lin_mov[app_fast] OR lin_mov[app_med]) OR lin_mov[app_slow]) AND zone
    [not]) AND fear[TRUE] THEN action[hesitancy]
IF (lin_mov[ret_fast] OR lin_mov[ret_med]) AND fear[TRUE] THEN action[
    escape]
IF lin_mov[ret_slow] AND fear[TRUE] THEN action[shock]
IF ((lin_mov[ret_slow] OR lin_mov[ret_med]) OR lin_mov[ret_fast]) AND
    disgust[TRUE] THEN action[refuse]
IF lin_mov[still] AND disgust[TRUE] THEN action[hesitancy]
IF ((lin_mov[app_slow] OR lin_mov[app_med]) OR lin_mov[app_fast]) AND
    disgust[TRUE] THEN action[perplexity]
IF neutral[TRUE] THEN action[none]
```

List of Figures

List of Tables