Neural Style Transfer with Twin-Delayed DDPG for Shared Control of Robotic Manipulators

Raul Fernandez-Fernandez[1*], Marco Aggravi[2], Paolo Robuffo Giordano[2], Juan G. Victores[1] and Claudio Pacchierotti[2]

***Abstract*— Neural Style Transfer (NST) refers to a class of algorithms able to manipulate an element, most often images, to adopt the appearance or style of another one. Each element is defined as a combination of Content and Style: the Content can be conceptually defined as the "what" and the Style as the "how" of said element. In this context, we propose a custom NST framework for transferring a set of styles to the motion of a robotic manipulator, e.g., the same robotic task can be carried out in an "angry", "happy", "calm", or "sad" way. An autoencoder architecture extracts and defines the Content and the Style of the target robot motions. A Twin Delayed Deep Deterministic Policy Gradient (TD3) network generates the robot control policy using the loss defined by the autoencoder. The proposed Neural Policy Style Transfer TD3 (NPST3[3]) alters the robot motion by introducing the trained style. Such an approach can be implemented either offline, for carrying out autonomous robot motions in dynamic environments, or online, for adapting at runtime the style of a teleoperated robot. The considered styles can be learned online from human demonstrations. We carried out an evaluation with human subjects enrolling 73 volunteers, asking them to recognize the style behind some representative robotic motions. Results show a good recognition rate, proving that it is possible to convey different styles to a robot using this approach.**

***Index Terms*— Style Transfer, Deep Reinforcement Learning, TD3, Autoencoders, NPST.**

I. Introduction

Neural Style Transfer (NST) was proposed by Gatys *et al.* [1] to define the Content and Style of an image using neural networks. Gatys *et al.* used features extracted by a VGG-19 pre-trained neural network [2] to define the Content as the elements represented in the painting (e.g., people, animals, houses), and the Style as the low-level features specific to the artist (e.g., vivid colors, long brushstrokes).

One advantage of NST is the definition of a high-level layer of abstraction that allows the same definition of Content and Style in a larger range of applications. One example is the area of motion animation [3]. Here the Content can be defined as the movements performed by the animated figure (e.g., moving forward, walk in circles, move sideways) and the Style as the emotion conveyed by these movements (e.g., sad, happy, tired, angry). Holden *et al.* [3] proposed the introduction of autoencoders for working with motions. The result is a NST generated motion trajectory that can be transferred to the animated figure. To the best of our knowledge, such an approach has never been applied to robotic autonomous control and teleoperation.

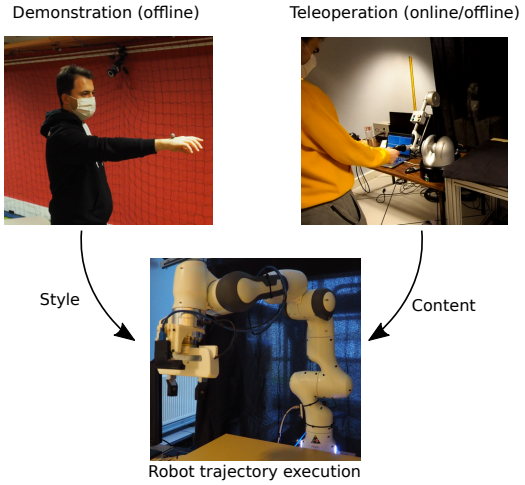

Fig. 1. Using Neural Style Transfer (NST), we can alter a robotic motion (the Content) according to the Style of a pre-recorded human demonstration. A certain teleoperated robotic motion can be carried out in, e.g., a angry, happy, calm, or sad way.

Previous works of the authors [4] have proposed an initial approach for introducing NST within discrete action spaces using Q-Networks. In this paper, we propose the use of a Neural Policy Style Transfer Twin Delayed Deep Deterministic Policy Gradient (NPST3) framework for introducing NST in continuos robotic action spaces, with teleoperation, and autoencoders to extract the Style Transfer loss. DRL has already been used to generate robust robotic controllers [5], while Deep Deterministic Policy Gradient (DDPG) [6] is an application of DRL within continuous action spaces. Finally, Twin Delayed DDPG (TD3) [7] has shown promising results as an advanced and more reliable approach to DDPG.

Our objective is to abstract the Style of a human motion and then apply it to the Content of an unrelated robotic motion. The latter can be generated offline by an autonomous controller or online by a human operator teleoperating the target robot. Autoencoders are introduced to define the Content and Style for training, while the TD3 algorithm is used to generate the robot control policies and encode the Style Transfer optimization step. The Content is defined as the high-level features that define the robot action (e.g., end-point of the trajectory) while the Style is defined as the low-level features specific to a certain human demonstration (e.g., speed, jerkiness). As a proof of concept, we consider four different Styles defined via human demonstrations: angry, happy, calm, and sad motions. Similarly, we define the

[1]Robotics Lab, Department of Systems Engineering and Automation, Universidad Carlos III de Madrid (UC3M), e-mail: {rauferna, jcgvicto}@ing.uc3m.es.

[2]CNRS, Univ Rennes, Inria, IRISA – Rennes, France, e-mail: {marco.aggravi, prg, claudio.pacchierotti}@irisa.fr.

[*] Corresponding author: rauferna@ing.uc3m.es.

[3]NPST3: Neural Policy Style Transfer TD3

 DOI: 10.1109/ICRA46639.2022.9812245

Content as the control motion imparted by a human operator in teleoperation. Doing so, a human user teleoperates a robot that, in turn, actuates the overall imparted motion in a angry, happy, calm, or sad way, locally deviating from the teleoperated Content motion to apply the chosen Style. The framework idea is depicted in Fig. 1.

Being able to impart a certain Style to a robotic motion via NST opens a new area of possible applications and can be of interest for art performances, animatronics, robot caregivers and waiters in smart city applications [8], craftsmanship, and in any other situation where a personalized motion is somehow important. An implementation of our approach, including all the statistics parameters we considered, is available at https://github.com/RaulFdzbis/NPST3.

II. Background and Preliminaries

Style Transfer is used to extract the Content and Style of the motions, while DRL permits to execute and generate the control policies defining the new robotic motions.

A. Style Transfer

The introduction and modification of Style in motions is not a new topic within the computer animation community [9]. The first works introducing Style in the area of animation proposed signal processing techniques for the definition and extraction of the Style [10]. More recent works proposed the implementation of more advanced techniques as multilinear model design [11]. The first works to differentiate between Content and Style, however, were part of the area of optical character recognition [12]. In robotics, Style Transfer has been related to the introduction of emotions in robotic motions. These works include the introduction of cost functions [13] or Laban movement systems [14].

With the publication of NST by Gatys *et al.* [1], a new layer of abstraction was introduced through the introduction of the VGG-19 pre-trained classification neural network. Due to the difficulty of having a proper motion classification pre-trained neural network, Holden *et al.* [3] proposed the introduction of autoencoders as an alternative. Autoencoders are neural networks that can be divided in two parts: the encoder and the decoder. The encoder is composed by the first layers of the network in charge of extracting the relevant features to generate the encoded data. An encoder operation can be defined following $A(X) = ReLU(\Psi(X * W_0 + b_0))$, where X is the input, Ψ is the pooling operation posterior to the first layer, W_0 is the weight matrix of the encoder, b_0 is the layer bias, and $ReLU$ is a Rectified Linear Units (ReLU) [15] activation. The decoder, corresponding to the last layers, takes the encoded data and performs the encoder inverse operation to regenerate the input. The Content loss can then be defined as the difference in the encoder outputs when passing the content and generated motions following

$$L_{content} = \|A(C) - A(G)\|, \tag{1}$$

where C and G are the Content and Generated motions, respectively. The Style loss is defined as the difference between the Gram matrix Gm of the encoder outputs when passing the style motion and Generated motions as in

$$L_{style} = \|Gm(A(S)) - Gm(A(G))\|, \tag{2}$$

where S is the Style motion. Finally, the total Style Transfer loss is defined as the sum of these two losses $L_{st} = L_{content} + L_{style}$.

B. Deep Reinforcement Learning

Robotics tasks usually require continuous and high-dimensional action spaces, while standard Reinforcement Learning techniques, like Q-learning [16], can only be directly applied to discrete action spaces. To address this limitation, Lillicrap *et al.* [6] proposed the introduction of DDPG for continuous action spaces. In this approach, the policy is defined as a parametric function. This parametric function is trained to maximize the output of the Q-value function. The authors proposed the introduction of two different neural networks within an actor-critic architecture to define the policy and the Q-value function. Based on this idea, TD3 was later proposed by Fujimoto *et al.* [7] as a more stable version of DDPG. TD3 improves the performance of DDPG by introducing Double Q-learning [17], target policy smoothing, and delayed policy updates.

III. Framework

The framework is depicted in Fig. 2. The Style Transfer block with the autoencoder network (blue) defines the Style Transfer loss, while the execution block with the TD3 policy network (green) minimizes the obtained Style Transfer and constraints losses. After training, the resulting TD3 policy network encodes the Style Transfer step, and it is the only network required for the control of the robot (red).

A. Inputs

The inputs of the framework are three 3D Cartesian motion trajectories that define the Content, the Style, and the Generated motions. The total length of all the input trajectories is fixed to 5 seconds. The number of samples per second is set to 10. These settings are a trade-off between motion accuracy and computational cost, as the motions need to be Generated at runtime. The motion trajectory matrices (C,S,G) used as inputs, have a $[m, n]$ shape, where m is the total number of samples, in this case 50, and n is the space dimension, in this case 3. The framework is designed to be able to work with incomplete motions (trajectories defined with less than 50 samples) allowing the Generated and Content motions to be used as input while being generated.

B. Autoencoder network: the loss network

A convolutional autoencoder network defines the loss network. Following the architecture proposed by Holden *et al.* [3], the encoder is defined using a 1D convolutional layer followed by a pooling operation layer. The convolutional layer is composed by 256 nodes and a kernel size of 5. The two layers of the decoder are defined as the transpose layers of the convolutional and pooling layers of the encoder. For the encoder, an additional dropout layer is introduced to improve the performance. The resulting network is used to extract the content and style loss as in Eq. 1 and 2.

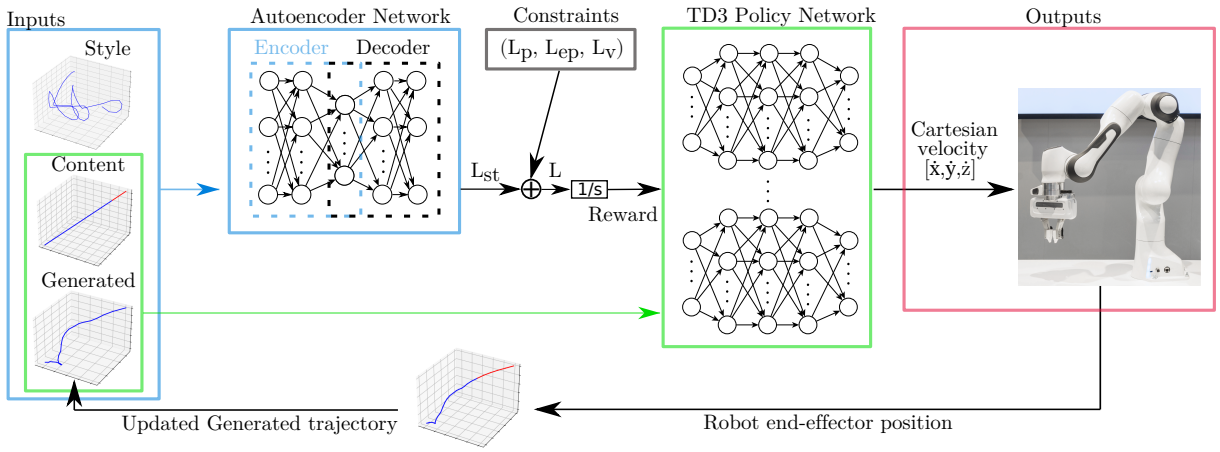

Fig. 2. Framework for the NPST3 algorithm. The Style, Content, and Generated motion trajectories are used as input for the autoencoder network. The loss obtained with the autoencoder network (L_{st}) is added to the resulting constraint loss (L_p, L_{ep}, L_v) as to obtain the overall loss L. The inverse (1/s) of this loss is the reward used for training the TD3 Policy network, together with the Content and Generated trajectories as input. The result of the TD3 network is a 3D Cartesian velocity vector, which is executed by the robot end effector. Finally, the Generated trajectory is updated adding the new position actuated by the robot, while the Content trajectory is updated using the positions defined by the user. The Content trajectory can be defined online via teleoperation or offline via a preplanned motion.

C. Constraints

Similarly to Holden *et al.* [3], in addition to the loss defined by the loss network, a few additional constraints are introduced to generate feasible and acceptable motions for the robot, e.g., the generated trajectory should be as similar as possible to the one commanded by the operator, there should no be discontinuities between trajectory executions, or the velocity of the generated trajectory should be bounded.

The first constraint limits the position error of the Generated motion with respect to the Content motion.

$$L_p = \| \frac{G[t-1] - C[t-1]}{RT} \|, \tag{3}$$

where t is the current execution step and the Robot Threshold (RT) is a handcrafted constant to normalize the values of the Cartesian workspace, assumed the same for all axes.

The second constraint limits the position error introduced on the last step of the motion trajectory. This error is introduced to smooth the transition between consecutive motions. This constraint is introduced following

$$L_{ep} = \| \frac{G[t_n] - C[t_n]}{RT} \|, \tag{4}$$

where t_n is the last time step. This loss is set to 0 if the last time step has not been reached.

The third and last constraint is a velocity constraint introduced to increase the weight of the velocity for the Generated motion. This constraint is defined as

$$L_v = \| (\frac{\partial G}{\partial t} - \frac{\partial S}{\partial t}) / RT \|. \tag{5}$$

The total loss is defined as a weighted sum of these constraints and the Style Transfer losses as in

$$L = w_c L_{content} + w_s L_{style} + w_p L_p + w_{ep} L_{ep} + w_v L_v \tag{6}$$

This loss will be used later for training the execution network.

D. TD3 Policy network: the execution network

The execution network is a TD3 network in charge of the generation of the Generated motion that minimize the loss defined in the Style Transfer stage. The inputs of the network are the Content and Generated motion trajectories. Both trajectories are incrementally introduced to the network. This allows the online generation of the Content motion. We need to train a different network for each style; however, all of these style-tailored networks will work for any Content.

The TD3 architecture is composed by two different sub-networks: the actor and the critic [18]. The actor network encodes the control policy of the robot, while the critic encodes the Q-value function. For the actor network, each input passes through three 1D convolutional layers before being concatenated with the other. The first convolutional layer is composed by 256 nodes and a kernel size of 5. This layer was designed to have the same configuration than the one used in the encoder. The rest of the convolutional layers have 128 nodes and a kernel size of 5. After these three layers, the two outputs of these convolutional layers are flattened and concatenated. The concatenated output is passed through four fully connected layers. The first two layers have a total of 512 nodes, the third 400, and the last one 300. A batch normalization layer is added between all the layers of the network. A ReLU activation is used for all the layers except the last one which uses an hyperbolic tangent (tanh) activation. The output of the network is a 3D velocity vector of the robot end-effector.

The critic network has the same architecture as the actor network with some exceptions. First, an additional input is introduced for the action, defined as the output of the actor network. This action is not passed through the convolutional layers, but instead directly through the fully connected layers. After the first fully connected layer, the action is concatenated with the two other inputs. An additional fully connected layer of size 512 is added before the 400 size layer. Finally, the last layer of the network uses a linear

activation function and its output corresponds to the Q-value. The specific design choices reported have been chosen during pilot experiments.

E. Outputs

The output of the framework is a 3D Cartesian velocity vector. This velocity vector is used to command the end effector of the selected robotic platform, e.g., a robotic arm in our case. The new end effector position, obtained after the execution of the velocity vector, is used to update the Generated motion trajectory. This updated Generated motion trajectory is used as input for the next step of the framework. This approach can be used with any robotic platform that can be controlled through velocity commands.

IV. Training

Similarly to the work proposed by Li *et al.* [19], to train the autoencoder we used the CMU Graphics Lab Motion Capture Database [20]. This dataset contains various motions performed by different people, captured using a motion capture system. For simplicity, only the information from the tracker at the end of the right hand (*RFIN*) was considered.

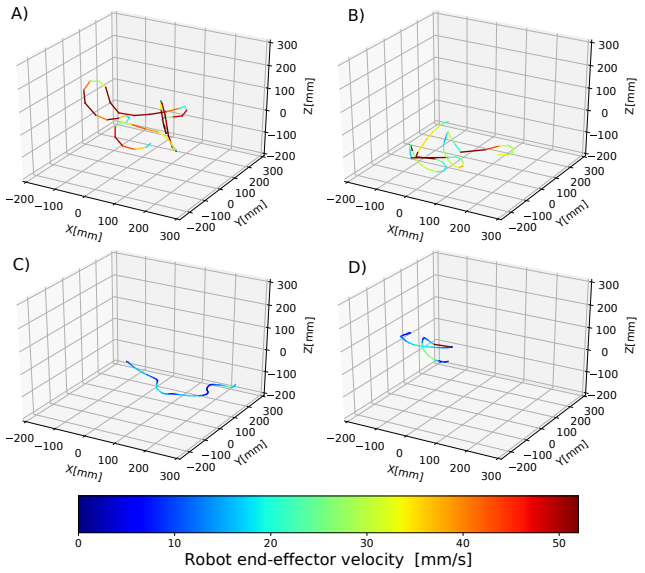

Fig. 3. Cartesian trajectories of the selected Styles. A) anger/annoyance, B) happiness/joy, C) calm/acceptance, and D) sadness/grief. The units in the axis are in mm.

In the case of the execution network, a random generator of linear motions was implemented to generate the training Content motions. We considered four Styles, related to the emotions of anger/annoyance, happiness/joy, calm/acceptance, and sadness/grief, which are part of the wheel of basic emotions as defined by Plutchik [21]. Such Styles were defined via human demonstrations: a single person was asked to move his right hand freely in space, so as to convey – in any ways he wanted to – each of the four emotions mentioned above. The position of the user's hand over time was recorded using a Vicon motion capture system. For the training, a 5-seconds-long portion of the full demonstrated motion was selected for each Style. Four

TABLE I
Training Parameters

Hyperparameters	Values
Shared	
Motion input shape	[50,3]
Sample frequency	10 [Hz]
Motion length	5 [s]
Motion Robot Threshold (RT)	300 [mm]
Optimizer	Adam [22]
Number of styles	4
Autoencoder	
Epochs	1000
Batch size	256
TD3 network	
Action space dimensions	3
Action Range (AR)	$\pm$ 0.1 * RT
Loss weights $(w_c, w_s, w_p, w_{ep}, w_v)$	(100, 1, 0.1, 1, 20)
Epochs	2500
Experience Replay size	10e3
Batch size	64
Critic Learning Rate	1e-5
Actor Learning Rate	1e-6
Discount (γ)	0.99
Critic/Actor update ratio	2
Target update value (tau)	1e-3
Loss Function	Mean Squared Error
Initialization network values	$\pm$ 3e-3 (Uniform)
Policy noise	0.002 * RT (Normal)
Action noise	0.02 * RT (Normal)

execution networks were then trained, each corresponding to one Style. Finally, a TD3 algorithm was implemented for the training. The loss described by Eq. 6 was selected as the loss function to minimize. The training parameters of the two networks are listed in Table I. They were defined after a set of preliminary experiments consisting of 50 tentative trainings, retaining the parameters best expressing the target emotions as identified by the user who carried out the demonstrations.

V. Experiments

A human subjects experiment was proposed for testing the performance of the NPST3 framework.

A. Subjects

We enrolled 73 volunteers (43 males, 29 females, 1 prefer not to say; age from 18 to 76 years old). Subjects came from different parts of the world (10 nationalities, 4 countries of residence), they had different education backgrounds (from middle school diplomas to PhD) and working situation (active, unemployed, and retired). Only few of them (12) had an experience in the robotics field. Subjects were able to carry out the experiment remotely.

B. Methods and Task

We considered a representative simple Content motion, a straight line, depicted at the top of Fig. 4. Then, we applied the four considered Styles following the methods described in Sec. III, yielding to four different motions. For the sake of simplicity, the motions were generated offline, but the framework is capable of generating them at runtime, e.g., a user teleoperates a robotic manipulator and the chosen Style is applied immediately to the motion of the robot. The four resulting motions were executed by a 7-DoF Franka Emika Panda robot, commanding velocities to the robot with respect

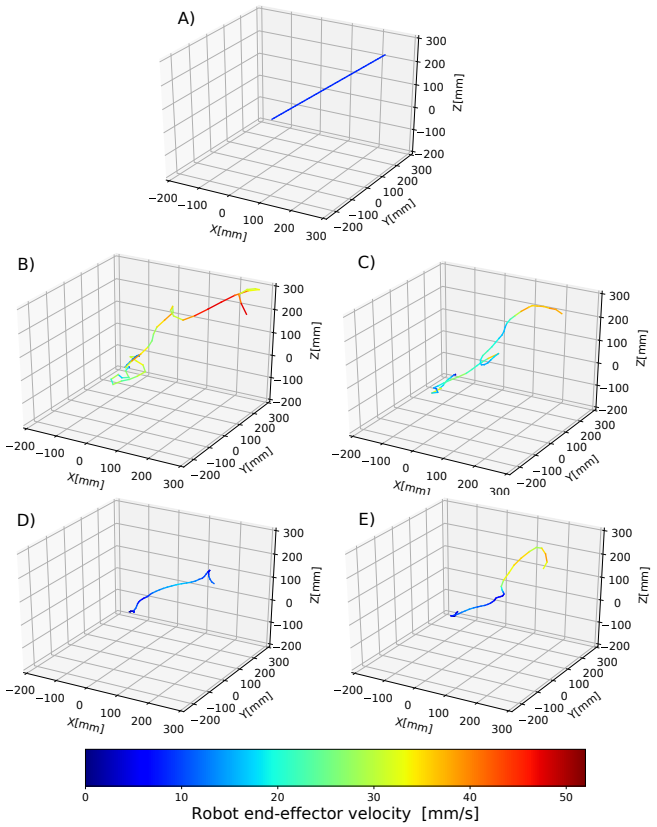

Fig. 4. Cartesian motions generated with the NPST3 algorithm. The trajectory at the top A) depicts the Content motion. The transferred Styles are: B) anger/annoyance, C) happiness/joy, D) calm/acceptance, and E) sadness/grief. The style trajectories are extracted from the right hand of a human demonstrator using a Vicon optical motion capture system.

to its end-effector frame. Each motion was recorded through an external camera posed in front of the robot, resulting in four videos showing the same Content trajectory executed by the Franka robot in the four different Styles. The video of these four executions is available as supplemental material and at https://youtu.be/ynfx2hhfoUs.

Subjects were presented with the four videos, with no information regarding the considered Styles nor about how the proposed framework works. For each video, subjects were asked to write down, using one word, what kind of emotion the robot motion elicits in them. Once their choices were registered, they were asked to watch the videos again. This time, subjects were asked to match each video with one of the considered Styles, i.e., anger/annoyance, happiness/joy, calm/acceptance, and sadness/grief, using a dropdown menu next to each video. Subjects could watch the videos how many times they wanted.

C. Results

Results are summarized in Figs. 5, 6 and 7. Figs. 5 and 7 shows the results of the first part of the questionnaire (free text response). In Fig. 5, the responses are placed within a wheel of emotions [21]. Each answered emotion is given a coordinate and the average response is reported in bold. We can see that users answered with emotions very close to the target one. In Fig. 7, emotions are ordered as a function of

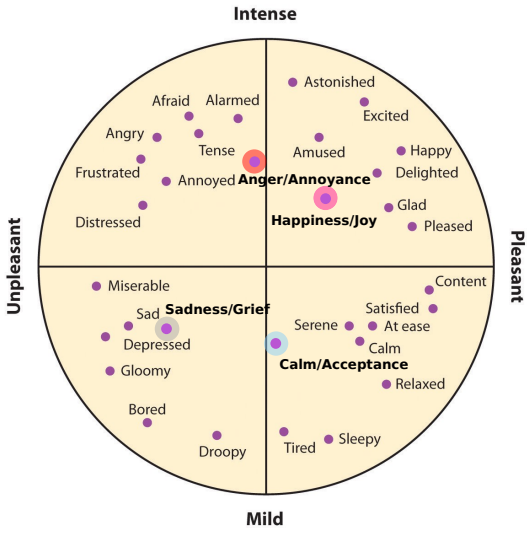

Fig. 5. Wheel of emotion. Considering the center of the wheel as the origin of a standard Cartesian coordinate system growing towards "Intense" (top) and "Pleasant" (right), we can give to each emotion in the wheel a coordinate, e.g., "pleased" can be at (14, 3) and "tired" at (-15, 1). Doing so, we can calculate the average answers of the subjects and place them in the wheel (bold). Figure inspired from Foxcroft *et al.* [23] and based on the model proposed by Rusell *et al.* [24].

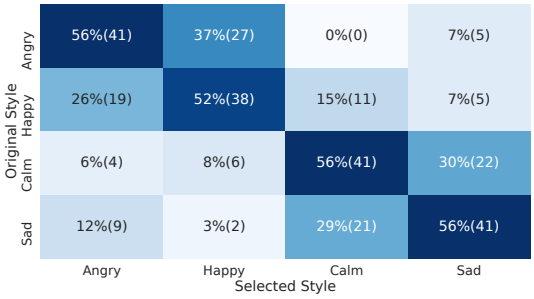

Fig. 6. Constrained multiple choice response. Each row depicts a different Style and associated video shown to the participants. Each column depicts the selected Style. The cells contain the percentage and number of times each Style was selected. All the videos were presented to all the volunteers.

the frequency in which they appear. Finally, Fig. 6 shows the results of the second part of the questionnaire (constrained multiple choice response).

VI. Discussion and Conclusions

In this paper, NPST3 is proposed as a way to introduce Style Transfer in continuos robotic action spaces using DRL. The resulting network is designed to work with incomplete motion trajectories. This allows the online teleoperation of the robot during the Style Transfer process.

The results obtained with the experiments show how the proposed NPST3 framework is able to successfully transfer the original Style to the generated robotic motion. Results in Figs. 5, 6 and 7 show that subjects were quite effective in identifying the correct emotion, even in the first free-text response part. In the second part, the main confusion was between anger vs. happiness and calm vs. sadness. While most subjects identified correctly the emotion (always >50%), those who did not confused happy/angry and calm/sad.

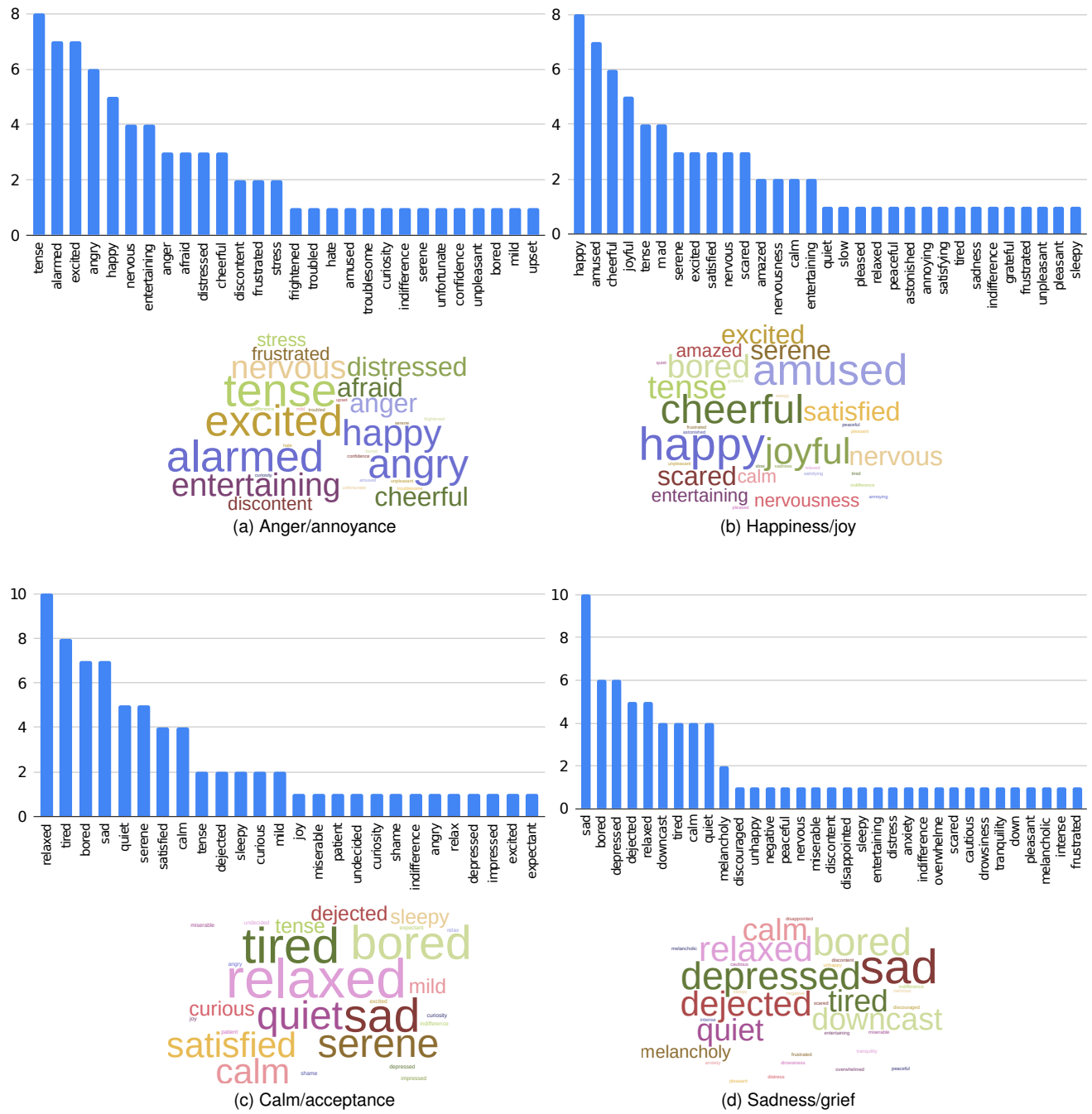

Fig. 7. Free text response. Each set of figures represents the results obtained with each of the four videos: (a) anger/annoyance, (b) happiness/joy, (c) calm/acceptance, (d) sadness/grief. The histograms show the frequency of the responses given by the subjects. For easier visualization, the same information is also reported using a word cloud (words in a larger font were more often chosen). The four motions were showed to all of the volunteers.

While there is still work to do in extending this approach to other types of robots (e.g., humanoids, mobile robots) as well as to other robotic motions (e.g., joint-by-joint Stylized motion), this work is, in our opinion, a promising starting point for introducing NST-based control in robotics. In the next future, we plan to implement the proposed approach on humanoids, studying how the motion can be modified using Style Transfer as well as how such motion can be related to other features of the robot, e.g., its facial expression.

VII. Acknowledgment

The research leading to these results has received funding from: RoboCity2030-DIH-CM, Madrid Robotics Digital Innovation Hub, S2018/NMT-4331, funded by “Programas de Actividades I+D en la Comunidad de Madrid” and cofunded by Structural Funds of the EU; ROBOASSET, ”Sistemas robóticos inteligentes de diagnóstico y rehabilitación de terapias de miembro superior”, PID2020-113508RB-I00 funded by AGENCIA ESTATAL DE INVESTIGACION (AEI); and “Programa propio de investigación convocatoria de movilidad 2020” from Universidad Carlos III de Madrid.

REFERENCES

[1] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, 2016, pp. 2414–2423.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR*, 2015.

[3] D. Holden, I. Habibie, I. Kusajima, and T. Komura, "Fast neural style transfer for motion data," *IEEE Computer Graphics and Applications*, vol. 37, no. 4, pp. 42–49, 2017.

[4] R. Fernandez-Fernandez, J. G. Victores, J. J. Gago, D. Estevez, and C. Balaguer, "Neural policy style transfer," *Cognitive Systems Research*, vol. 72, pp. 23–32, 2022.

[5] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[6] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2016.

[7] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80. PMLR, 2018, pp. 1587–1596.

[8] R. Fernandez-Fernandez, J. G. Victores, D. Estevez, and C. Balaguer, "Real evaluations tractability using continuous goal-directed actions in smart city applications," *Sensors*, vol. 18, no. 11, 2018.

[9] A. Bruderlin and L. Williams, "Motion signal processing," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '95, 1995, pp. 97–104.

[10] M. Unuma, K. Anjyo, and R. Takeuchi, "Fourier principles for emotion-based human figure animation," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '95. Association for Computing Machinery, 1995, pp. 91–96.

[11] J. Min, H. Liu, and J. Chai, "Synthesis and editing of personalized stylistic human motion," in *Proceedings of the 2010 ACM SIGGRAPH*

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations (ICLR)*, 2015.

symposium on Interactive 3D Graphics and Games, ser. I3D '10. Association for Computing Machinery, 2010, pp. 39–46.

[12] J. B. Tenenbaum and W. T. Freeman, "Separating style and content," *Advances in neural information processing systems*, vol. 9, 1997.

[13] A. Zhou and A. D. Dragan, "Cost functions for robot motion style," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3632–3639.

[14] M. Sharma, D. Hildebrandt, G. Newman, J. E. Young, and R. Eskicioglu, "Communicating affect via flight path exploring use of the laban effort system for designing affective locomotion paths," in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2013, pp. 293–300.

[15] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. Omnipress, 2010, p. 807–814.

[16] C. J. C. H. Watkins and P. Dayan, "Q-learning," in *Machine Learning*, 1992, pp. 279–292.

[17] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, p. 2094–2100.

[18] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds., vol. 12. MIT Press, 2000.

[19] Y. Li, Z. Wang, X. Yang, M. Wang, S. I. Poiana, E. Chaudhry, and J. Zhang, "Efficient convolutional hierarchical autoencoder for human motion prediction," *The Visual Computer*, vol. 35, no. 6, pp. 1143–1156, 2019.

[20] "CMU graphics lab: Carnegie mellon university motion capture database," 2003. [Online]. Available: http://mocap.cs.cmu.edu/

[21] R. Plutchik, "A psychoevolutionary theory of emotions," *Social Science Information*, vol. 21, no. 4-5, pp. 529–553, 1982.

[23] C. Foxcroft and C. Panebianco-Warrens, "Exploring the role of pianists emotional engagement with music in a solo performance," *SAMUS : South African Music Studies*, vol. 34-35, no. 1, pp. 459–498, 2015.

[24] J. Russell, "A circumplex model of affect," *Journal of Personality and Social Psychology*, vol. 39, pp. 1161–1178, 12 1980.