Dimitri: A low-cost compliant humanoid torso designed for cognitive robotics research

Ahmadreza Ahmadi*, Christopher Tatsch†, Fabrcio Julian Carini Montenegro†, Jun Tani*, Rodrigo da Silva Guerra†

*Dept. of Electrical Engineering, KAIST, Daejeon, 305-701, Korea
Email: tani1216jp@gmail.com

†Centro de Tecnologia, Universidade Federal de Santa Maria, Av. Roraima, 1000, Santa Maria, RS, Brazil
Email: rodrigo.guerra@ufsm.br

***Abstract*—This work presents Dimitri, an open-software & open-hardware robot torso equipped with modular low cost compliant joints, built to allow advanced research on human-robot interaction, force-aware object manipulation and environment exploration. The robot has 13 DOFs: 4 DOFs with series elastic actuators in each arm, 3 DOFs for roll, pitch and yaw of the waist, and 2 DOFs for head pan and tilt. Our main innovation is in the employment of a polyurethane based compliant spring system attached to conventional robotics servo motors, turning them into low-cost series elastic actuators (SEAs). To demonstrate the robot's capabilities in the context of cognitive robotics research and goal-directed behavior, we successfully employed a multiple timescale recurrent neural network (MTRNN), enabling the robot to reproduce combined prototypical movement patterns previously trained via interactive demonstration.**

***Index Terms*—humanoid robot, compliant joints, MTRNN**

I. Introduction

Traditional robots, such as the ones found in the industry, are designed to perform highly decoupled movements following pre-programmed trajectories, while rejecting environmental disturbances along the way. This is typically achieved, among other things, by the employment of very stiff joint mechanisms, heavy and rigid link structures and powerful actuators. More recently, however, advances in the field of human-robot interaction have increased the demand for safe force-aware actuators with active or passive compliance. Beside being intrinsically safer, lighter compliant mechanisms allow the development of more advanced manipulation algorithms that explore the coupling with the environment through force feedback and control.

This work presents Dimitri, an open-platform torso humanoid robot equipped with low-cost modular SEAs in its arm joints. Dimitri is designed with the goal of allowing research on fields related to next-generation human-robot interaction and compliant force-based object manipulation. This type of architecture opens new grounds for biologically inspired neurorobotics research by allowing a more human-like approach to object manipulation and interaction. The modular SEAs enable the robot to solve manipulating problems that require compliance, force feedback and control. The robot can explore the dynamics of the environment by sensing the constraints it imposes to the robot's arms measured via torque feedback. The control of the strength being applied to an object allows the robot to "feel" the object and understand movement constraints imposed by the environment.

In order to validate the platform in the context of neurorobotics, we chose to implement a test experiment using a multiple timescale recurrent neural network (MTRNN), which is an active topic of research by the authors of this work. The MTRNN architecture was originally proposed by one of our co-authors, and it was chosen due to its capacity of encoding dynamic movement patterns, learning lower level primitive skills and organizing these skills in higher level goal directed plans.

The rest of this paper is organized as follows: Section II presents technical specifications regarding Dimitri's hardware and software, and describes the methodology employed for the MTRNN experiment; Section III shows the results; and Section IV makes the final remarks.

II. Methodology

This section is divided in four parts. Part II-A reviews the robot's main specifications. Part II-B describes de series elastic actuators (SEAs) employed in the arm joints. Part II-C briefly describes the software architecture. Finally Part II-D describes the MTRNN experiment.

A. Robot Specifications

Dimitri, shown in Fig. 1, was developed to be an innovative and robust open-source and open-hardware robot. All source code is available on GitHub[1] and the mechanical design files are available upon request.

Our goal was to create a humanoid robot suitable for research on compliance control and capable of safe human-robot interaction while keeping the project simple and modular. The result is a very minimalist mechanical design that can be assembled and maintained even by researchers with little experience on mechanical hardware. The robot can be easily extended to match specific purposes, including custom accessories, such as adding end effectors, designing expressive faces and adding custom external body frame.

[1]https://github.com/tioguerra/Dimitri

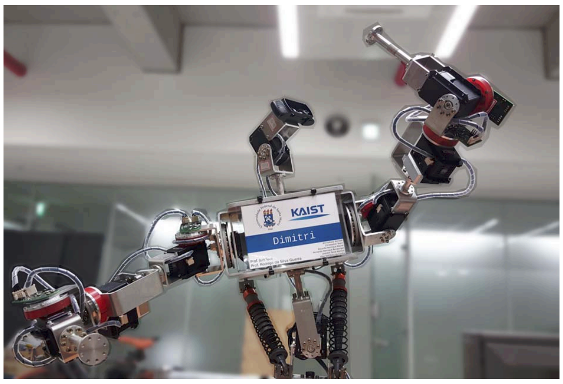

Fig. 1: Photograph of Dimitri.

Table I sumarizes the main features of the robot. The robot's structure is simple to manufacture and assemble. Figure 2 and Table II show the Denavit-Hartenberg parameters of this robot.

Dimitri's frame parts are made of 3mm thick aluminium sheets that are laser-cut and bent using a CNC bending machine. Only a few of the parts are milled in a CNC Lathe for aesthetic reasons. The series elastic actuators are composed of polyurethane springs and machined aluminium mounts equipped with a magnetometer based circuit board for measuring their angular displacements (see more about the SEAs in subsection II-B).

For the motors we chose the well known Dynamixel servo actuators manufactured by Robotis, using model MX-106R for arms and waist joints, and model MX-64R for neck. These actuators are networked together, along with the feedback circuit for the SEAs through a standard RS-485 bus using the Dynamixel protocol. For the computer we used an embedded NUC computer equipped with a 4-channel mPCI-e RS-485 board. The computer communicates with the motors and SEAs at 400kbps. These motors from Robotis allow torque control and combined with the polyurethane based compliant spring allow for force based modelling and manipulation. The motors provide up to 10N.m of stall torque, which is enough for the applications this robot was built to perform.

B. Modular series elastic actuator

The main feature of Dimitri is in its pair of arms, each equipped with compliant joints in their 4 DOFs. A Series Elastic Actuator [1] basically consists of a traditional servo actuator in series with a spring connected to the load. This allows the load to be partially decoupled from the motor, and the exerted force to be evaluated by measuring the spring's deflection. Our proposed SEA consists of a torsional spring designed to be attached to Dynamixel MX series servo actuators, manufactured by Robotis (for more details see [2], [3]). This servo actuator was chosen due to its wide popularity among researchers in the robotics field, however the general idea can be easily adapted to fit other servo actuators of similar "RC-servo-style" design. The elastic element consists of a two-part component, using a modular spring designed using a thermoplastic polyurethane (TPU) elastomer (see Figure 4). The material is cheap, tough, easy to mill and presents rubber-like elasticity [4]. This is an extremely low-cost design since it can be easily manufactured using a 3-axis CNC router. The shape of the spring was designed so as to allow large

Physical Specifications	
Robot Height	392.5mm
Reach	542.5mm
Robot Weight	5.719kg
Degrees of Freedom	13
Number of SEA Joints	8 (4 in each arm)
Max Payload	2.5kg
Servomotors	MX-106R and MX-64R
Frame material	Aluminium
Computer	
Processor	4th gen. Intel Core i5 4250U
Memory	8GB 1333MHz DDR3L
Storage	120GB SSD mSATA
Control bus	
PC interface	mPCI-e
Number of channels	4
Bus protocol	RS-485
Baudrate	400kbps
Camera	
Model	FireFlyMV FMVU-13S2C
Resolution	1280×960
Frame Rate	up to 60 FPS
Lens Mount	C / CS
Electrical Specifications	
Supply Voltage	12 Volts DC
Max Consumption	1200 Watts

TABLE I: Dimitri's Technical Specifications.

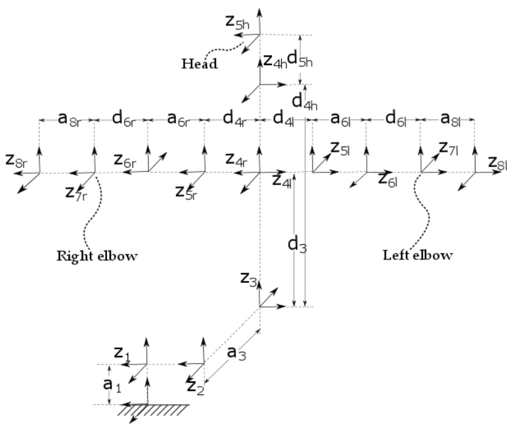

Fig. 2: Reference diagram for Denavit-Hartenberg parameters.

i	α_i	a_i	d_i	θ_i
0	0	0	0	θ_0
1	0	a_1	0	θ_1
2	$\pi/2$	0	0	θ_2
3	$-\pi/2$	a_3	0	θ_3
4	$\pi/2$	0	d_3	θ_4
5	$\pi/2$	0	d_4	θ_5
6	$-\pi/2$	a_6	0	θ_6
7	$\pi/2$	0	d_6	θ_7
8	0	a_8	0	0

TABLE II: Denavit-Hartenberg parameters right arm.

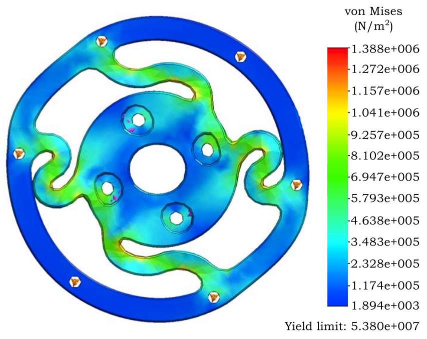

Fig. 3: Finite Element Analysis of von Mises stress.

Fig. 4: Manufactured polyurethane-based torsional spring.

angular displacements both clockwise and counter-clockwise, with a linear force/angle ratio over the full range of torque supported by the motors. Figure 3 shows the simulated spring displacement at maximum motor torque.

C. Software Structure

The base code for controlling motors and SEAs is released under the MIT open-source license. The code is object-oriented, written in C++ with minimal dependencies. Except for the vision processing, which depends on OpenCV, all base code depends only on standard system libraries.

The UML class diagram is drafted in Figure 5. Class `JointChain` combines series of joints (instances of `Joint`) to compose the arms, the neck and the waist. These `JointChain` objects are brought together in the main class `Dimitri`. Class `ElasticJoint` derives from class `Joint`, extending its functions to include the feedback from the springs and a PID controller.

D. MTRNN Experiment

A MTRNN is a type of recurrent neural network that uses neurons with different timescales capable of self-organizing a functional hierarchy. Neurons with a small time constant are called fast context (FC) units, and ones that have a larger time constant are called slow context (SC) units. Similar to [5], input and output units are only connected to FC neurons. In our work all output units have time constant of 1 and show no recurrent connections. The current external input states can be received by the input units and their predicted states are generated by the output units.

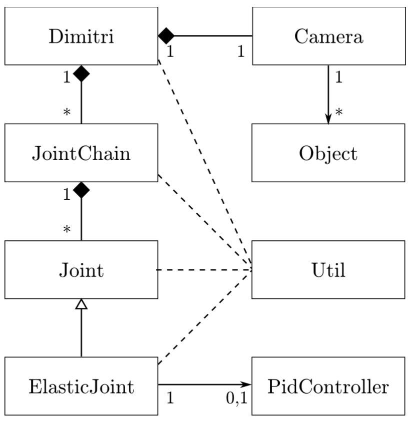

Fig. 5: Simplified UML class diagram.

Neural activities are calculated based on a conventional firing rate model in which each neurons activity entails average firing rate of other neurons, and its own decayed internal value from the previous time step as shown in 1

$$u_{i,t+1} = \left(1 - \frac{1}{\tau_i}\right) u_{i,t} + \frac{1}{\tau_i}\left(\sum_j w_{ij} c_{j,t} + \sum_j w_{ik} x_{k,t} + b_i\right) \tag{1}$$

where $u_{i,t}$ is the internal state value of i_{th} neuron at time t, w_{ij} is the connectivity weight from i_{th} context neuron to j_{th} context neuron, w_{ik} is the connectivity weight from i_{th} neuron unit to k_{th} input unit, $c_{j,t}$ is the context activation value of j_{th} neuron at time t, $x_{k,t}$ is the external input of k_{th} input unit at time t, b_i is the bias of the i_{th} neuron, and τ_i is the time constant of the i_{th} neuron. If the neuron belongs to SC units, the second summation term does not exist, since there are no connections to the input units. The connection weights between context neurons (w_{ij}) are bidirectional, and all neurons are connected to each other.

1) Softmax: A softmax transformation is used to remap each input $y_{i,t}$ into a higher dimensional space $y_{ij,t}$ according to receptive fields of adjacent intervals of equal length. The transformation is performed as follows [6]

$$y_{ij,t} = \frac{\exp\left(\frac{-||k_{ij} - j_{i,t}||^2}{\sigma}\right)}{\sum_{j \in Z} \exp\left(\frac{-||k_{ij} - j_{i,t}||^2}{\sigma}\right)} \tag{2}$$

where k_{ij} represents the j_{th} neuron of the reference vector for the i_{th} dimension of the real input, $y_{i,t}$ is the i_{th} dimension of the real input value before transformation at time t, σ is a constant value that specifies the shape of the distribution (0.05 in our experiment), and $y_{ij,t}$ is the transformed vector. Equation below computes the reference vectors

$$k_{ij} = B_i^{Min} + \frac{B_i^{Max} - B_i^{Min}}{l(i) - 1}(j-1) \tag{3}$$

where B_i^{Min}, B_i^{Max}, and *l(i)* represent the minimum value for i_{th} dimension of the real input data, the maximum value for i_{th} dimension of the real input data, and dimension of the reference vector, respectively.

The dimension of the reference vector in this study is 11, which means that this transformation changes each real input dimension to 11. The transformation is applied to each input dimension independently. Our experiment used 10 real input dimensions, resulting in 110 softmax input dimensions in which each 11 of them are computed independently.

The inverse softmax transformation described below is used to calculate the i_{th} dimension of the real output units:

$$y_{i,t} = \sum_{j \in Z} y_{ij,t} k_{ij} \tag{4}$$

2) Generation and training methods: As shown in equation 1, the internal dynamics of the context neurons can be obtained based on a conventional firing rate. To obtain the context output values for the forward dynamics, the activation function was $c_{i,t+1}$ = (1.7159)$\tanh$(0.666667$u_{i,t+1}$). The following equations perform the forward dynamics of the output unit

$$u_{ij,t+1} = \sum_{l} w_{ijl} c_{l,t} + b_{ij} \tag{5}$$

$$y_{ij,t+1} = \frac{\exp{(u_{ij,t+1})}}{\sum_k \exp(u_{ik,t+1})} \tag{6}$$

where $u_{ij,t+1}$ is the internal state of the j_{th} softmax output unit corresponding to the i_{th} real output unit, w_{ijl} is the connectivity weight from the l_{th} neuron in the FC units to the j_{th} softmax output unit corresponding to the i_{th} real output unit, $c_{l,t}$ is the cotext output of the l_{th} neuron, b_{ij} is the bias of the j_{th} softmax output unit corresponding to the i_{th} real output unit, and $y_{ij,t+1}$ is the j_{th} softmax output unit corresponding to the i_{th} real output unit at time *t+1*.

A conventional back-propagation through time (BPTT) scheme is used for the network training [7]. The learnable parameters are optimized in the direction of minimization of the Kullbak-Leibler divergence (noted as E) between desired and real activation values of softmax output units ($\bar{y}_{i,t+1}$ and $y_{i,t+1}$, respectively), according to the equation

$$E = \sum_t \sum_{i \in O} y_{i,t+1} \log(\frac{\bar{y}_{i,t+1}}{y_{i,t+1}}) \tag{7}$$

All learnable parameters, noted as θ, are weights and biases and initial states that approach to their optimal values in the opposite direction of the gradient $\frac{\partial E}{\partial \theta}$, and they are updated as follows

$$\nabla\theta(n+1) = \mu\nabla\theta(n) - \alpha\frac{\partial E}{\partial \theta} \tag{8}$$

$$\theta(n+1) = \theta(n) + \nabla\theta(n+1) \tag{9}$$

where α is the learning rate, set as 0.00003 in our experiment, and μ is the momentum term, set 0.9 in our experiment. The equations about the weight, bias and initial states gradients can be seen in [8].

All learnable parameters are set randomly at the beginning of the training from a uniform distribution on the interval [-$\frac{1}{M}$, $\frac{1}{M}$], where M is the number of context neuron units.

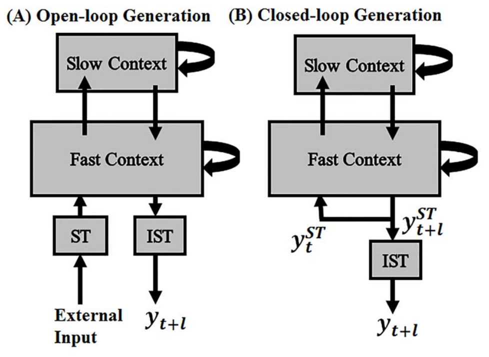

Fig. 6: chemes for (A) open-loop generation (B) closed-loop generation. ST and IST are the abbreviations for the softmax transform, and inverse of the softmax transform, respectively.

In of our experiment, the open-loop generation approach is used in the training mode in which MTRNN receives the current external inputs and generates one or multiple look-ahead prediction steps of the outputs. The closed-loop mode is defined as giving the current prediction outputs to the next time inputs. This can also be referred as the mental simulation of actions [9], [10]. Schematics of open-loop and closed-loop generations are shown in Figure 6(A) and 6(B), respectively. In Fig. 6, l represents the number of look-ahead prediction steps of the output that the MTRNN generated based on the current input.

III. Results

We employed MTRNN to work as the brain of our robot, allowing Dimitri to learn some combined prototypical movement patterns and later generate them in the testing phase. For this purpose, we trained three combined patterns by grasping the robot's hand and demonstrating the desired movements. A green cubical object (6×6×6 cm) was put on a table in front of Dimitri and Dimitri's camera was tracking the object by changing the neck's pan and tilt angles – this task was done by using the OpenCV framework included in the base code. The neck joint angles were considered as our visual inputs. To collect the training data, the robot always started from the same home position. After initialization, the torques of both arm joins (8 DOFs) were set to low values while the waist joints were fixed with a high torque. This made it easy for the experimenter to move the robot's arms in order to demonstrate the compositional movement patterns.

Three prototypical patterns were defined: (1) Push object with right hand (labeled PUSH); (2) Touch object with right hand (labeled TOUCH); and (3) Hit table with alternating hands (labeled HIT). Instead of training these in independently, they were trained through compositional patterns, which consisted of a combination of two successive prototypical patterns demonstrated in sequence. The compositional patterns were:

(A) HP[2] / PUSH / HP / HIT / HP

[2]HP stands for Home Position

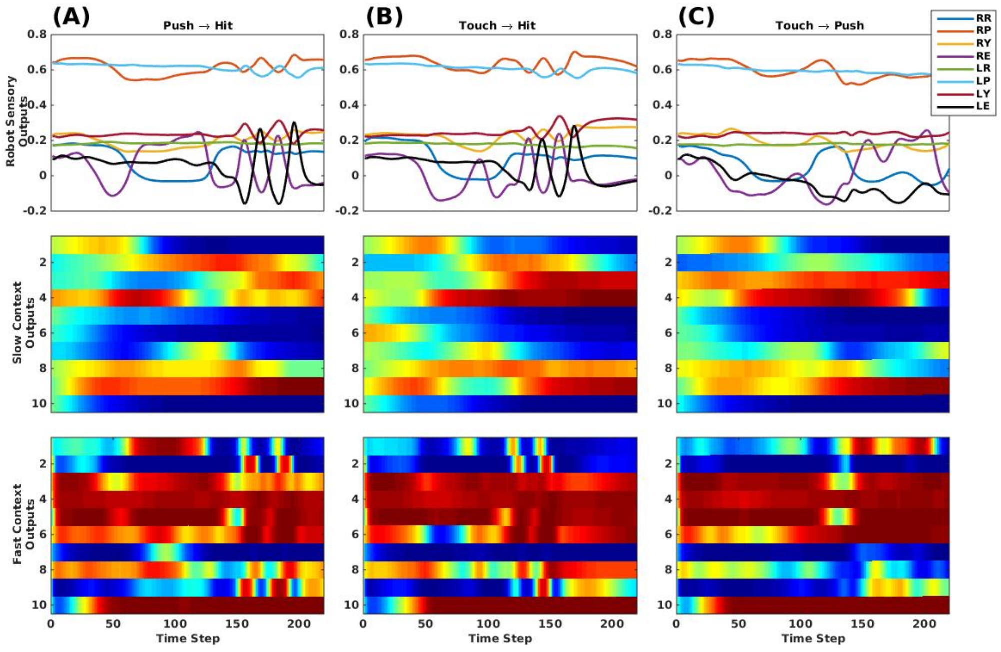

Fig. 7: Results of the MTRNN experiment for combined prototypical movement patterns. RR: Right Arm Roll, RP: Right Arm Pitch, RY: Right Arm Yaw, RE: Right Arm Elbow, and similarly for left arm with LR, LP, LY and LE.

(B) HP / TOUCH / HP / HIT / HP
(C) HP / TOUCH / HP / PUSH / HP

Three training data sequences were demonstrated and recorded for each compositional pattern by starting with the green object on three different y coordinate positions (middle of the table; 9cm above; 9cm below), resulting in a total of 9 training sequences. After this, the recorded sequences were used to train the MTRNN model, which consisted of 40 FC, 20 SC, 110 softmax input and 110 softmax output units derived from 10 dimensions (8 for the arms and 2 for the neck) as described in II-D1. The time constants of FC and SC units were set to 5 and 100, respectively[3]. Only the SC initial states were updated during training, while FC initial states were always set to zero. The initial states of slow context neurons had different values for each training sequence. In other words, through an association between initial states and corresponding sequences, MTRNN could be trained to generate multiple sequences. The network was trained for 85000 epochs to generate 5 look-ahead prediction steps ($l = 5$) of the next input sequences using open-loop generation.

After the training was successfully finished, the MTRNN was used to control Dimitri's arms. Dimitri's waist joints were fixed and the neck angles were commanded to track the object, giving the resulting values as external visual inputs to the network. The MTRNN generated the corresponding arms joint angles by means of the closed-loop generation. This means that MTRNN was in a semi-closed loop state (visual inputs were in an open-loop mode, while arms joint angles were in a closed-loop mode). We tested the MTRNN for the three initial object positions and neck joint angles for all 9 compositional movement patterns. Dimitri was able to generate all 9 combinatorial movement patterns successfully. Figure 7 (A), (B), and (C) display the results for one of the initial y coordinate positions. The transitions of primitives patterns can be observed on the figures of the first row, which show the arms joint angles over time. The second row illustrates the SC neural activities of 10 selected neurons, and the third row shows the FC neural activities of 10 selected neurons. Visual inputs (neck joint angles) are not displayed for the sake of clarity. It can be seen that the self-organization of a functional hierarchy is successfully performed since their periodicity are different in a way that SC activities are changed slowly to observe the transitions among prototypical movement patterns while FC activities are changed fast in order to encode the specific dynamics of each primitive movement pattern. We also performed the mental simulation of the combinatorial prototypical patterns using totally closed-loop generation with no external inputs by means of the initial states of the SC units. Dimitri succeeded to generate all 9 patterns.

IV. Conclusion

This work presented Dimitri, an open-software and open-hardware humanoid torso robot designed for research on mechanical compliance and its applications. Our goal was to present a low cost alternative suitable for researchers in low-income countries. Despite its limitations, such simplistic design easily proves its value when the task requires joint compliance and safe human-robot interaction. The MTRNN

[3]Unit is in number of samples.

results showed that different compositional patterns could be generated by the aid of the initial sensitivities of the SC units in both closed-loop and semi-closed loop manners, and the intention could be changed in a top-down manner from the higher neural states to the output units. Experiments involving online learning and exploration are not only performed with more safety, but also include torque feedback.

The springs were found to perform very well, adding force sensibility with very little compromise on joint speed. The aluminium frame proved itself robust and light enough. One noticeable disadvantage is the susceptibility for oscillatory behaviour. More present on shoulder joints, oscillations were easily controlled through PID gain adjustments, at the expense of the system's response speed.

Besides providing compliance, force feedback and control for dynamic object manipulation, Dimitri's modular SEAs help the joints to absorb shock, making the robot very robust to impact. As an experiment to illustrate this robustness we dropped a brick of 2.57kg from 50cm above the robot's left arm, as shown in Figure 8. The elastic elements bent during the impact and the control over the servo motors quickly recovered the arm back to its original position.

For future work we plan on extending the capabilities of the robot in order to allow research on themes such as bipedal walking and human social interaction. We are currently manufacturing a biped leg design based on parallel link mechanisms combined with our SEAs for the knee joints. We are also developing a prototype animatronic face. Furthermore, wrist and end effector extension designs are being tested.

References

[1] G. Pratt and M. Williamson, "Human robot interaction and cooperative robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 399 – 406, 1995.

[2] L. T. Martins, R. d. S. Guerra, C. Tatsch, E. H. Maciel, and R. Gerndt, "Polyurethane-based modular series elastic upgrade to a robotics actuator," in *Proc. of the Robot Soccer World Cup XVIII*, 2015.

[3] L. T. Martins, R. d. S. Guerra, R. Gerndt, E. H. Maciel, and C. Tatsch, "A polyurethane-based compliant element for upgrading conventional servos into series elastic actuators.," in *Proc. of the 11th IFAC Symposium on Robot Control*, 2015.

[4] M. Ashby and K. Johnson, *Materials and desing: the art and science of material selection in product design*. Rio de Janeiro: Elsevier Editora Ltda, 2011.

[5] Y. Yamashita and J. Tani, "Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment," *PLoS Comput Biol*, vol. 4, no. 11, p. e1000220, 2008.

[6] C. M. Bishop, "Pattern recognition," *Machine Learning*, 2006.

[7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., DTIC Document, 1985.

[8] G. Park and J. Tani, "Development of compositional and contextual communicable congruence in robots by using dynamic neural network models," *Neural Networks*, vol. 72, pp. 109–122, 2015.

[9] M. Jeannerod, "The representing brain: Neural correlates of motor intention and imagery," *Behavioral and Brain sciences*, vol. 17, no. 02, pp. 187–202, 1994.

[10] J. Tani, "Model-based learning for mobile robot navigation from the dynamical systems perspective," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 26, no. 3, pp. 421–436, 1996.

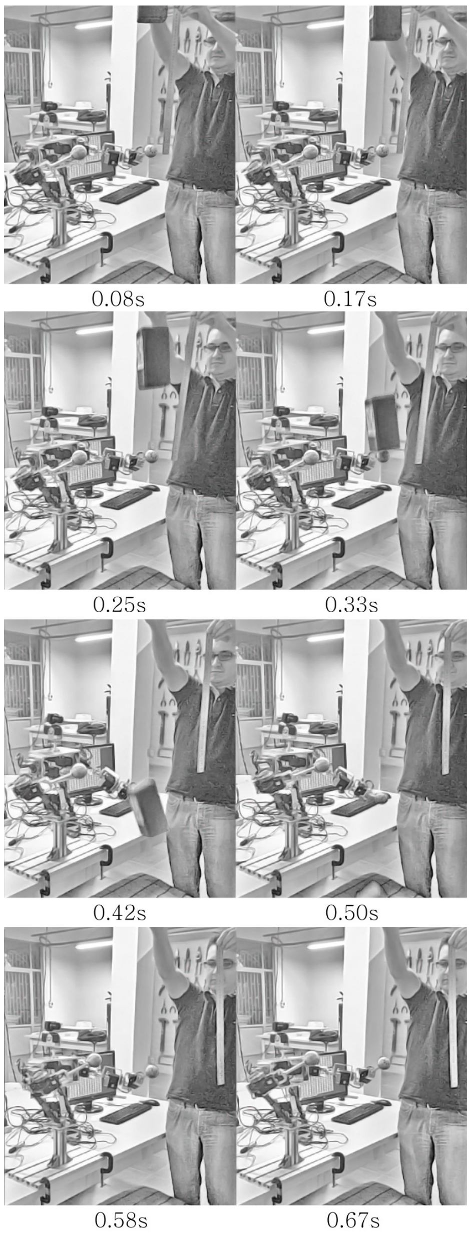

Fig. 8: A 2.57kg brick is dropped from a height of 50cm over Dimitri's left arm, causing no damage. The robot recovers the original posture less than a second later.