Positioning Control on a Collaborative Robot by Sensor Fusion with Liquid State Machines

Davi Alberto Sala
and Valner João Brusamarello
Electrical Engineering Department
Universidade Federal do Rio Grande do Sul
Porto Alegre, RS, Brasil
Email: {davi.sala,valner.brusamarello}@ufrgs.br

Ricardo de Azambuja
and Angelo Cangelosi
School of Computing, Eletronics and Mathematics
University of Plymouth
Plymouth, United Kingdom
Email: {ricardo.deazambuja,a.cangelosi}@plymouth.ac.uk

***Abstract*—A positioning controller based on Spiking Neural Networks for sensor fusion suitable to run on a neuromorphic computer is presented in this work. The proposed framework uses the paradigm of reservoir computing to control the collaborative robot BAXTER. The system was designed to work in parallel with Liquid State Machines that performs trajectories in 2D closed shapes. In order to keep a felt pen touching a drawing surface, data from sensors of force and distance are fed to the controller. The system was trained using data from a Proportional Integral Derivative controller, merging the data from both sensors. The results show that the LSM can learn the behavior of a PID controller on different situations.**

***Index Terms*—Robot control, Sensor Fusion, Liquid State Machine, BAXTER robot, PID controller.**

I. Introduction

Multi-sensor data fusion is a largely disseminated subject [1] employed in several fields, such as surveillance, controlling robots, driving autonomous vehicles and others. In typical applications, redundant and complementary data sensors are blended in order to produce robust systems by increasing their reliability. Generally, data fusion involves data association, state estimation, and or decision fusion, by applying mathematical tools such as Kalman filter [2], Bayesian networks [3], fuzzy logic [4] and neural networks, among others [5].

Many applications employ neural networks. In [6], where multi-sensor data are used as input and the system is trained for monitoring of tool wear in drilling and milling. Also, the authors of [7] present a model for sensory fusion using an Echo State Network to predict object trajectories based on the information from several radars. Other works such as [8], [9] present models for sensor fusion using artificial neural networks. Those works featured good results in their applications, demonstrating that processing information from several sensors using artificial neural networks can considerably enhance process identification and monitoring.

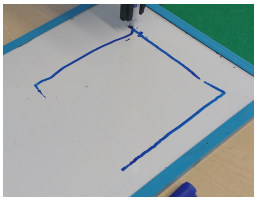
The human brain is commonly used as inspiration when working with artificial neural networks, mainly for its characteristics of robustness and power efficiency. While a modern computer in a common configuration needs a power supply of about 500W, our brain spends only 20W [10]. In neuromorphic engineering [11], devices are inspired by the human brain. Several projects based on neuromorphic computers are currently in development, such as SpiNNaker [12], Darwin [13], IBM-TrueNorth [14] and Spikey [15]. All of them are based on Spiking Neural Networks (SNN), the artificial neuron model that better represents the way our neurons work.

Researchers from University of Plymouth developed a model [16] based on parallel, diverse and noisy sets of biologically inspired Liquid State Machines (LSM) [17] to control BAXTER [18], a collaborative humanoid robot, with the intent of creating a framework that could later be imported into a neuromorphic computer. The system is able to learn the trajectories necessary to draw a two dimensional shape on the surface of a table. However, this work does not have a fine Z-axis control, since the proprioceptive feedback comes only from the joint angles, and any changes on surface, robot leveling or external noise introduce errors into the drawings.

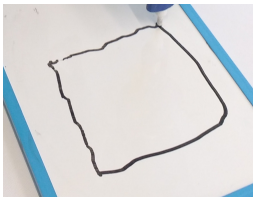
In this paper, we propose a parallel control LSM framework to work with the main robot control loop in order to reduce the positioning error of the pen. The proposed system can learn the behavior of a classic Proportional Integral Derivative (PID) controller, maintaining a constant distance from the table and applying a stable force and, therefore, keeping the pen touching the table.

Experimental data showed that the controller was able to significantly reduce the error previously observed at tracking the square. For instance Figs. 1(a) and 1(b) present the results of two drawn squares without the proposed Z-axis control system on the Z-axis. In Fig. 1(a) the felt pen does not touch the surface in some parts of the trajectory and touches too hard in Fig. 1(b), causing a distortion on the drawing. Fig. 1(c) shows the trajectory of a square performed with the implemented LSM control on Z-axis. In this case, the felt pen was neither floating nor dragging, which is the expected result.

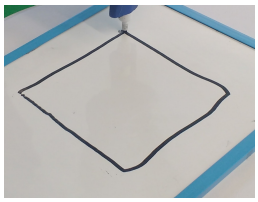
Although the example is a simple square trajectory, theoretically the system could learn, and if necessary correct, any 2D trajectory. The trajectory control system presented



(a) Low force (no control)



(b) High force (no control)



(c) Stable force (LSM control)

Fig. 1. BAXTER Robot drawings on a hard surface (white board). On (a) and (b) no Z-axis control is used, (c) presents our proposed LSM controller.

by Azambuja [16] also performed a triangle and circular trajectory.

II. Methodology

The BAXTER robot was commanded using the Robot Operating System (ROS) [19], the LSMs were created using the BEE[1] library and joint trajectories were calculated using the Virtual Robot Experimentation Platform (V-REP) [20]. This project was developed using, and the entire source code is available online[2].

A brief summary of the methodology that was applied is shown below:

1) The LSM training set is created using two PID controllers;
2) N random liquids are created;
3) Training data is converted into train of spikes and feed into the created liquids;
4) Readout weights for each of the N random liquids are calculated;
5) Evaluate how each LSM performs individually through real-world experiments using the BAXTER robot.

The training set was produced using a felt pen on a soft surface, on three different configurations: one without any inclination, one inclined by $5mm$ on X axis and by $5mm$ on Y axis, as displayed in Fig. 2. The Liquid State Machines were individually evaluated on soft surface (a notebook) and a hard surface (white board - Fig. 1), both situations with and without inclinations.

A. Liquid State Machines

A Liquid State Machine (LSM) [21] consists of neurons randomly and recurrently connected ("the liquid"), to which

[1]Source code available at https://github.com/ricardodeazambuja/LiquidStateMachine-Python

[2]Source code and data sets available at https://github.com/davi-sala/I2MTC2017-LSMFusion

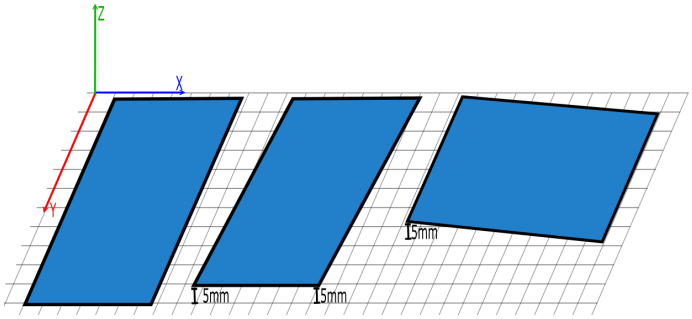


Fig. 2. Dispositions of the drawing surface for the creation of the training set. Elevation of $5mm$ was used on the X and Y axis, separately, resulting in an angle of $4°$ between the surface and table.

time series of spike patterns are sent. The liquid, or reservoir, can be seen as a random function of the input.

The concept of LSM [17] can be explained as a reservoir of recurrently iterative nodes, stimulated by an input $u(t)$. A state $x(t)$ is obtained from the reservoir state and a reading function f^M maps the high dimensional state $x(t)$ to an output $y(t)$. Fig. 3 shows a diagram of this architecture. The loops created between nodes of the reservoir create a form of short-term memory, where the effect of the inputs remain within the network for a certain amount of time according to the liquid parameters..

The LSM used in this work and in the robot's main control loop [16] employed the same parameters and neuron models from [22] with a few modifications. However, our framework does not make use of Short-Term Plasticity (STP) or forced transmission delays.

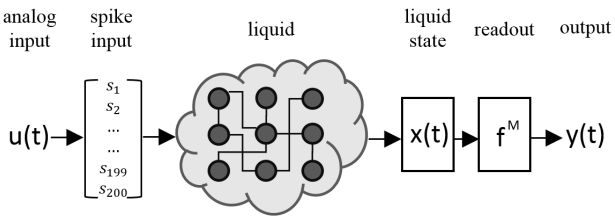


Fig. 3. Diagram showing a model of Liquid State Machine. An input is translated into a train of spikes, which stimulates the liquid layer. A readout function maps the liquid state $x(t)$ to an output $y(t)$.

B. Square Trajectory

The collaborative robot BAXTER, from Rethink Robotics Inc., is a safe robot designed to work among humans. The developed LSM framework to control BAXTER in [16] was designed using Joint Position Control[3]. This mode allows all the robot's safety mechanisms to be activated during movement execution. In the main control loop, four joints (S0, S1, E1 and W1) are controlled, pen orientation is kept perpendicular to the surface and the trajectory is divided in 1000 steps. Joint angles for the square shape were generated

[3]See http://sdk.rethinkrobotics.com/wiki/Arm_Control_Modes

using inverse kinematics on the V-REP simulator, keeping a constant Z-axis position.

The BAXTER robot uses Series Elastic Actuators (SEAs) [23] introducing a spring between the motor elements and the output of the actuator. This results in higher safety, as the springs can be deformed by human level inputs, but adds noise to the output. The SEAs also enable us to measure the torque output from the actuators, which allows to estimate force on each joint and also at the endpoint (felt pen in this case). Alongside the noise introduced by the actuators, an outside spring supporting S1 joint (see Fig. 4) also add interferences on the arm movement.

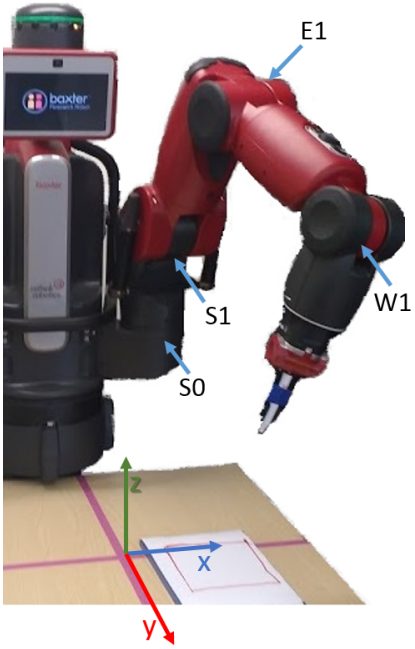


Fig. 4. Disposition of the BAXTER robot to perform the task. The four joints, with names highlighted, and the orientation of the coordinate system.

The noise introduced by BAXTER's actuators results in Z-axis positioning errors when moving the arm. Fig. 5 shows the Z position registered by the robot, this value is estimated by the robot's internal model using joint positions. While the V-REP simulation returned a fixed value at $-160mm$ on the Z-axis, as expected since the simulated BAXTER does not have any kind of noise modeled, the value for the real robot oscillates around the average value $\mu = -159mm$ (values obtained in 10 trials) with a standard deviation of $\sigma = 1.27mm$. The minimum and maximum peaks observed in figure 5 were generated by movements of joint S1.

C. Generating the Training Data

Since a training set is necessary to find the readout function of the LSM (see [21] for details), two PID controllers were implemented to create a dataset containing 100 trials, each with 1000 points of the square trajectory.

PID is a controller widely used for error reduction and faster response in process control. The PID parameters were adjusted

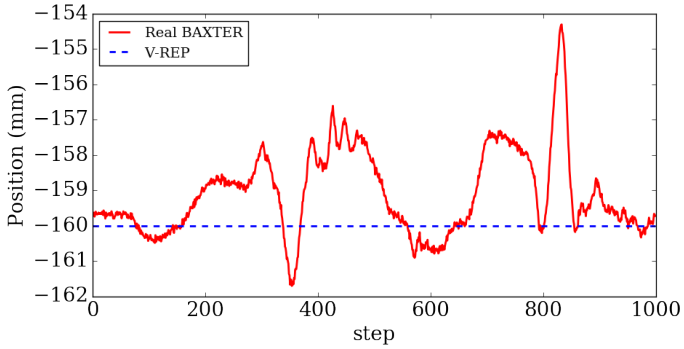


Fig. 5. Z-axis distance measured by the robot's built-in model when performing the same square shape trajectory on simulator and real world.

using the Ziegler-Nichols method [24]. Fig. 6 shows how the PIDs were inserted to the control system: one PID was utilized to keep the range sensor at a setpoint of $15.7cm$, the second was set to keep the force at $-2N$. A negative value, as the estimated force points towards the robots endpoint. The PIDs compensates the Z positioning offset error.

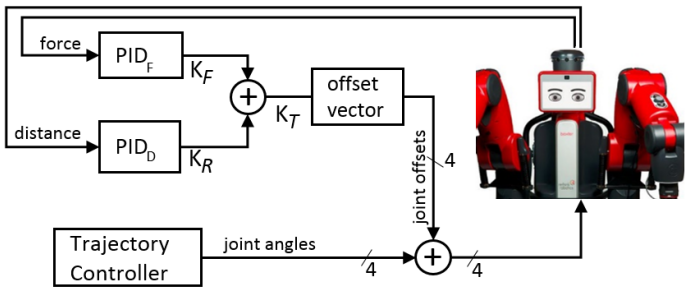


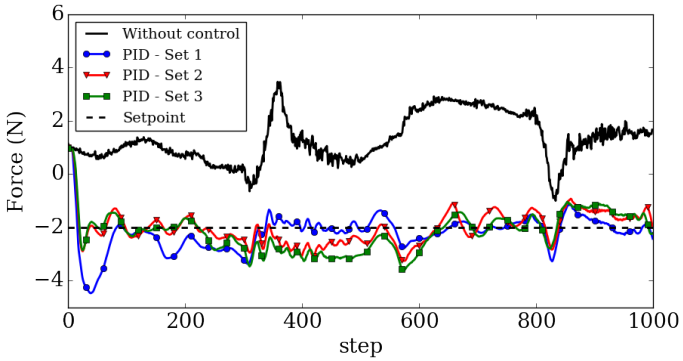
Fig. 6. Diagram of the PID system used to generate the training set. Outputs of the PIDs are summed and then multiplied compensated for each joint as an offset.

The training set consists of 100 trials, divided in 50 trials on a $0°$ surface, 25 trials with an inclination of approximately $4°$ on the X axis and the last 25 with the same inclination on the Y axis, all drawn on a soft surface (see Fig. 2). The trajectory was designed on V-REP with $1.0cm$ above the drawing surface. The pen would not touch the surface at any point by sending only the joint angles to the robot.

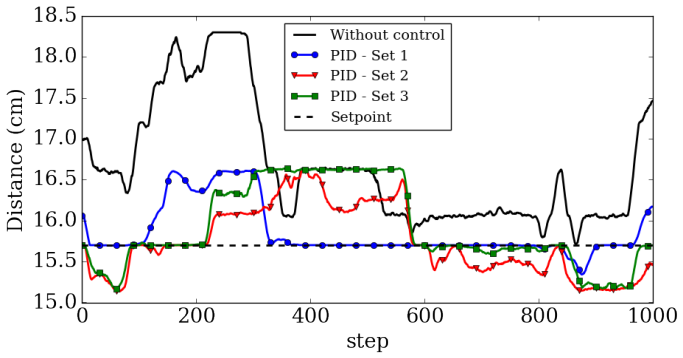
Average values of force, distance and final gains of the 100 trials using the PID controllers are showed in figure 7. The average value of 10 trials without the controller is also shown in Figs. 7(a) and 7(b).

D. Liquid State Machine Generation

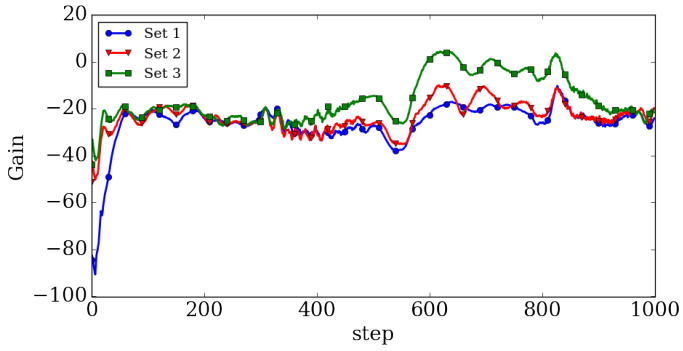
Five liquids were created, each with 200 neurons, 100 for each sensor input. These values were heuristically chosen after initial pilot tests, not presented here. A simplified population code was used to generate the spike inputs by discretizing analog values, the same algorithm that was used in [16]. As the number of neurons is limited, each input signal was conditioned between $-10N$ to $2N$ and $15cm$ to $18cm$, for force and distance sensors, respectively. The neuron model



(a) Force sensor readings using the PID controller



(b) Range sensor readings using the PID controller



(c) Summed gain of the PIDs controllers

Fig. 7. Values of the training set for force (a), distance (b) and gains (c). Blue, green and red lines represent the average value of 50 trials at 0°, 25 at 4° on the X axis, and 25 with 4° on the Y axis, respectively. Black lines on (a) and (b) represent the average values of 10 trials without the controller, and dashed lines show the setpoints used for each PID.

and LSM parameters used in this work are the same as used to control BAXTER in [16].

The liquids were simulated by receiving the discretized input signals from the training dataset, containing the PIDs trials. The output of the liquid's membrane low-pass filter is saved at each simulation step. This produces a matrix of size 1000×200 for every dataset element (100).

In order to train the weights (w_1 to w_{200}) of the readout function, a linear regression was implemented. The Ordinary Least Squares (OLS) fits a model with coefficients $w = (w_1, w_2, ...w_p)$ by solving the problem in (1), where matrix $\mathbf{X}$ contains the low-pass filtered values of the liquid spikes and y the final gain from the PIDs controllers for every trial.

$$\min_w ||\mathbf{X}w - y||_2^2 \quad (1)$$

When data are correlated, the columns of $\mathbf{X}$ are approximately linearly dependent. Consequently, the least-square estimate becomes highly sensitive to random errors, producing a large variance. The Ridge regression [25] was performed for every liquid to find the corresponding LSM readout weight vector, imposing a penalty ($\alpha \geq 0$) on the size of the coefficients, as can be seen in (2). Even if $\mathbf{X}$ is not full rank, the regression is still solvable.

$$\min_w ||\mathbf{X}w - y||_2^2 + \alpha||w||_2^2 \quad (2)$$

III. Results

The goal of this study was to minimize the variability on the endpoint by creating a LSM controller, enabling the robot to maintain a constant force against the drawing surface and decreasing the position error of the felt pen.

Results presented in [16] showed a difference of $1.93mm$ between the minimum and maximum Z-axis distance readings, when performing the square trajectory on the V-REP simulator. However, when using the same networks on the real robot, we measured fluctuations close to $6mm$. The trajectory used to validate our system presented a difference of approximately $7.6mm$ (see Fig. 5). The presented LSMs were evaluated on the real robot, by running the same square trajectory on different situations, changing parameters such as surface type and inclination. From the five created LSMs, the one that showed the better overall performance was selected and results are presented here. The values of average and variance from the LSM controller output were compared to the PID controller results.

Fig. 8 shows the average value of force from 50 experiments using the PID controller on the soft surface and the average value of 50 trials with the proposed LSM controller, on a hard surface. Mean values of 10 trials without any control are also presented for comparison. Through a simple visual inspection it is possible to observe that the LSM solution proposed in this work is able to follow the behavior of a PID controller.

Individual results of each data set were also analyzed. Table I shows the average values and standard deviations of the resulting force from all the trials performed using one LSM controller. Last column of the table shows the percentage of values that were found within the 95.44% Confidence Interval (CI) of the PID controller force data, considering a gaussian distribution. This comprehends all the force values measured between $-3.763N$ and $-0.567N$. Last row presents results of all the trials combined.

Considering that the felt pen would not be touching the surface when estimated force was above $0N$. From all trials, $11,12\%$ of force values were found above $0N$, i.e. not touching the table.

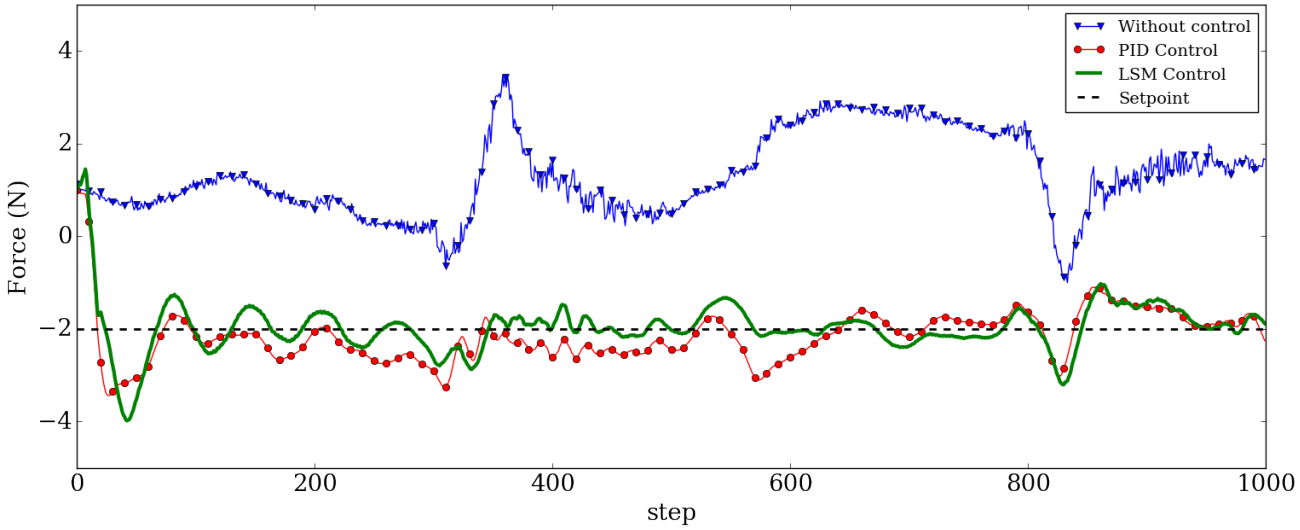


Fig. 8. Mean values of force sensor readings on three situations: without any control (blue curve), using the PID controller (red dashed curve) on a soft surface and LSM controller (green curve) on a hard surface. The PID setpoint is represented by the blue dashed line.

TABLE I
FORCE RESULTS OF LSM CONTROLLER PERFORMING IN FOUR DIFFERENT TEST SETS.

Test set	Mean value μ_i	Standard dev. σ_i	Values within PID's CI (95%)
Soft surface at 0°	$-1.392N$	$1.666N$	73.52%
Hard surface at 0°	$-1.982N$	$1.566N$	87.42%
Hard surface at $4°onY$	$-3.881N$	$3.348N$	62.14%
Hard surface at $-4°onX$	$-2.648N$	$1.773N$	75.38%
Combined	$-2.162N$	$2.418N$	74.15%

Both LSM and PID controllers performed the trajectory similarly, as one can see on Fig. 9, on the average distance values. Table II presents the average values and standard deviations of the distance measured by the Infrared distance sensor. Last column also shows the percentage of values found between the 95.44% CI of the PID controller data, considering a gaussian distribution. This includes all distance values ranging from $15.43cm$ to $16.21cm$.

TABLE II
DISTANCE RESULTS OF LSM CONTROLLER PERFORMING IN FOUR DIFFERENT TEST SETS.

Test set	Mean value μ_i	Standard dev. σ_i	Values within PID's CI (95%)
Soft surface at 0°	$15.79cm$	$0.51cm$	97.09%
Hard surface at 0°	$15.87cm$	$0.28cm$	97.87%
Hard surface at $4°onY$	$15.83cm$	$0.35cm$	97.82%
Hard surface at $-4°onX$	$15.88cm$	$0.33cm$	97.56%
Combined	$15.83cm$	$0.41cm$	97.49%

The LSM, on average, produced similar results as the PID controller, but presented a high variance on force readings. While the PID showed an average force value of $\mu_{PID_f} = -2.165N$ and standard deviation of $\sigma_{PID_f} = 0.799N$, the LSM controller displayed $\mu_{LSM_f} = -2.162N$ and $\sigma_{LSM_f} = 2.418N$. As for distance, readings from both approaches were similar, the PID controller presented an average distance of $\mu_{PID_d} = 15.82cm$ and standard deviation of $\sigma_{PID_d} = 0.39cm$ while the LSM controller displayed $\mu_{LSM_d} = 15.83cm$ and $\sigma_{LSM_d} = 0.41cm$.

IV. CONCLUSION

A framework based on Liquid State Machines, to control the Z-axis distance and force of a BAXTER robot, was presented in this paper. The proposed system was able to learn the behavior of a PID controller and showed to be able of assisting the robot to perform a trajectory on top of a flat surface, reducing the error by following a force and distance signals.

The trajectory used to benchmark the system here was the worst-case scenario. With an error of 100%, that is, the felt pen would never touch the drawing surface. Considering a Normal distribution, the presented system was able to correct the joints offset and keep within the PIDs 97.49% and 74.15% of the total points to the working distance and force respectively, inside a confidence interval of 95%.

Our approach was designed with the intent to work on a neuromorphic computer, in parallel to the main LSM control system. Future works would include the combination of both systems and the migration to the SpiNNaker neuromorphic system. We believe the accuracy could be improved by including different situations and trajectories in the training set. Further research on how a more complex training set can impact on the performance of the proposed system is another avenue to be pursue.

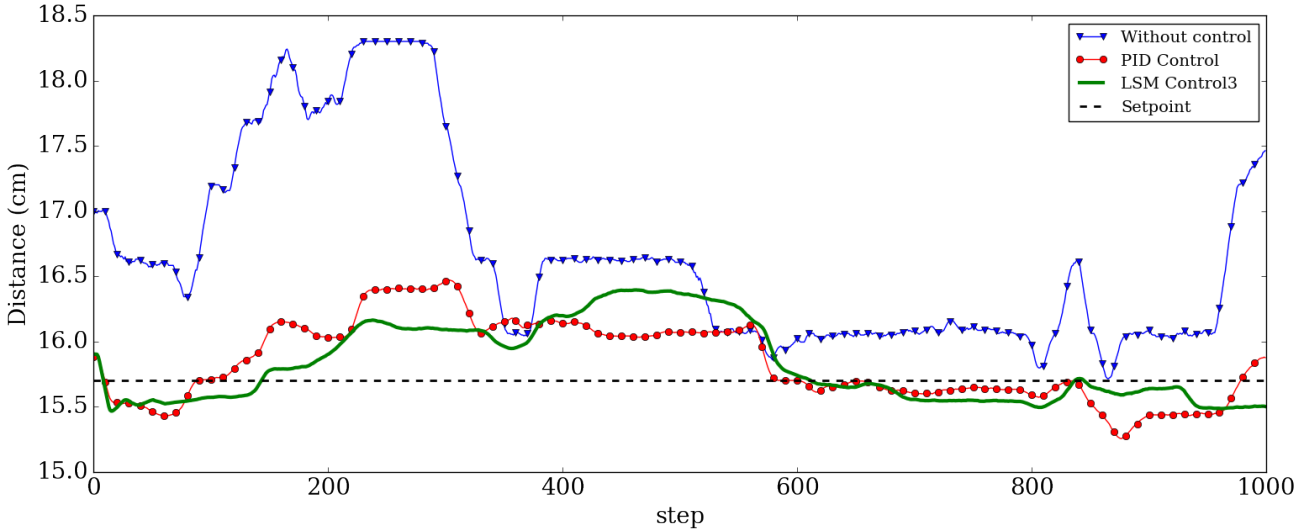


Fig. 9. Mean values of range sensor readings on three situations: without any control (blue curve), using the PID controller (red dashed curve) on a soft surface and LSM controller (green curve) on a hard surface, both without any inclination. The PID setpoint is represented by the blue dashed line.

ACKNOWLEDGMENT

This work was in part supported by CNPq, National Council for Scientific and Technological Development of Brazil (scholarship 131112/2015-5), CAPES Foundation, Ministry of Education of Brazil (scholarship BEX 1084/13-5) and UK EPSRC project BABEL (EP/J004561/1 and EP/J00457X/1).

REFERENCES

[1] M. E. Stieber, E. Petriu, and G. Vukovich, "Instrumentation architecture and sensor fusion for systems control," *IEEE Transactions on Instrumentation and Measurement*, vol. 47, no. 1, pp. 108–113, Feb 1998.

[2] D. Fontanelli, D. Macii, P. Nazemzadeh, and L. Palopoli, "Collaborative localization of robotic wheeled walkers using interlaced extended kalman filters," in *2016 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, May 2016, pp. 1–6.

[3] G. de Morais Borges and V. Brusamarello, "Bayesian fusion of multiple sensors for reliable heart rate detection," in *2014 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, May 2014, pp. 1310–1313.

[4] J. M. M. Villanueva, E. C. T. de Macedo, R. C. S. Freire, and E. C. Guedes, "Partial discharge measurement and uncertainty analysis based on fuzzy data fusion," in *Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International*, May 2012, pp. 1033–1038.

[5] R. R. Brooks and S. S. Iyengar, *Multi-sensor fusion: fundamentals and applications with software*. Prentice-Hall, Inc., 1998.

[6] I. Kandilli, M. Sonmez, H. Ertunc, and B. Cakir, "Online monitoring of tool wear in drilling and milling by multi-sensor neural network fusion," in *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, Aug. 2007, pp. 1388–1394.

[7] M. Li, B. Lv, W. Dong, and D. Wang, "Model of multi-sensor data fusion and trajectory prediction based on echo state network," in *Computer, Mechatronics, Control and Electronic Engineering (CMCE), 2010 International Conference on*, vol. 1, Aug. 2010, pp. 338–341.

[8] F. Garcia, P. Cerri, A. Broggi, A. de la Escalera, and J. Armingol, "Data fusion for overtaking vehicle detection based on radar and optical flow," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, Jun. 2012, pp. 494–499.

[9] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking deep belief network," *Frontiers in Neuroscience*, vol. 7, no. 178, 2013.

[10] D. Drubach, *The brain explained*. Prentice Hall, 2000.

[11] D. Monroe, "Neuromorphic computing gets ready for the (really) big time," *Commun. ACM*, vol. 57, no. 6, pp. 13–15, Jun. 2014.

[12] S. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the spinnaker system architecture." *IEEE Trans. Computers*, vol. 62, no. 12, pp. 2454–2467, 2013.

[13] J. Shen, D. Ma, Z. Gu, M. Zhang, X. Zhu, X. Xu, Q. Xu, Y. Shen, and G. Pan, "Darwin: a neuromorphic hardware co-processor based on spiking neural networks," *SCIENCE CHINA Information Sciences*, vol. 59, no. 2, pp. 1–5, 2016.

[14] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[15] T. Pfeil, A. Grubl, S. Jeltsch, E. Muller, P. Muller, M. A. Petrovici, M. Schmuker, D. Bruderle, J. Schemmel, and K. Meier, "Six networks on a universal neuromorphic computing substrate," *Frontiers in Neuroscience*, vol. 7, no. 11, 2013.

[16] R. Azambuja, A. Cangelosi, and S. V. Adams, "Diverse, noisy and parallel: a new spiking neural network approach for humanoid robot control," *International Joint Conference on Neural Networks*, pp. 1134–1142, 2016.

[17] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[18] C. Fitzgerald, "Developing baxter," in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, Apr. 2013, pp. 1–6.

[19] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[20] M. F. E. Rohmer, S. P. N. Singh, "V-rep: a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[21] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[22] P. Joshi and W. Maass, "Movement generation with circuits of spiking neurons," *Neural Computation*, vol. 17, no. 8, pp. 1715–1738, 2005.

[23] G. A. Pratt and M. M. Williamson, "Series elastic actuators," in *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, vol. 1. IEEE, 1995, pp. 399–406.

[24] K. Ogata and Y. Yang, *Modern Control Engineering Fifth Edition*. Prentice-Hall, 2010.
[25] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.