Humanoid motion representation by sensory state transitions

Fabio DallaLibera*, Takashi Minato†, Hiroshi Ishiguro†‡, Ademar Ferreira§, Enrico Pagello*, Emanuele Menegatti*
* Intelligent Autonomous Systems Laboratory, Department of Information Engineering (DEI),
Faculty of Engineering, University of Padua, Via Gradenigo 6/a, I-35131 Padova, Italy
† ERATO, Japan Science and Technology Agency,
Osaka University, Suita, Osaka, 565-0871, Japan
‡ Department of Adaptive Machine Systems,
Osaka University, Suita, Osaka, 565-0871 Japan
§ Departamento de Engenharia de Telecomunicações e Controle,
Universidade de São Paulo, São Paulo, 61548, Brasil

***Abstract*— Given the complexity and the high number of degrees of freedom humanoid robot motions often are generated off-line, stored in the robot memory and then reproduced. In the simplest cases motor commands are just replayed in open-loop (i.e. feed-forward), in more advanced implementations few parameters are changed on fly to adapt the motion to the external conditions. Indeed, several authors proposed a large variety of techniques to exploit the input from sensory feedback to modify a reference trajectory, in order to cope with environment changes and disturbances. In this paper, we propose a motion representation for humanoid robots that includes the sensory feedback information in the motion representation itself. This motion description allows stabilization against disturbances and environmental changes, but does not require any design or tuning of the relationships between sensory inputs and movement modification. We present experimental results on a simulated small humanoid robot equipped with motor encoders and touch sensors covering the whole body.**

***Index Terms*—Humanoid robot, robot motion, motion representation**

I. Introduction

Generation of humanoid robot motions is a complex task, due to the high number of degrees of freedom, usually 20 or more. Furthermore in the case of small humanoid robots the on board computing power is often quite limited so usually at least part of the motion is generated off-line. The most trivial approach consists of precomputing (by planning [1], optimization of parametric trajectories [2], [3], [4], motion retargetting of human motions [5] or design from scratch [6]) the joint trajectories or torques and replaying them ignoring sensory feedback [7]. Despite its disadvantages, feed-forward control is still largely employed for small humanoid robots, whose stability is usually improved by large footprints and a low center of mass position.

For tasks like walking [8], running [9] or crawling [10] specific techniques to exploit the sensory feedback in order to improve the robustness were developed. One commonly used approach is to modify the robot movement to control the Zero Movement Point [11], with the complexity of the controller ranging from the injection of torque into the ankle joints [12] to whole body movements [13]. These techniques allow robots which would otherwise be too unstable to be stabilized [14].

Another, biologically inspired approach consists of generating the motion using a dynamical system, termed as Central Patter Generator [15], [16]. Usually CPGs are obtained by the interconnection of oscillators, which in turn often consist of an extensor and a flexor neuron pair [15] that controls one of the actuators. In this case the motion is implicitly described by the weight of the connections between the neurons (appropriate weights are usually set automatically by the policy gradient [16] or genetic algorithms [15]). Adaptation to the environment is obtained by interconnecting, with proper weights, the sensory inputs to the neurons that generate the motion: this feedback entrains the robot in stable limit cycles that depend both on the robot structure and the environment. A recent, very appealing approach based on dynamical systems can be found in [17] while [18] shows how locally weighted regression can be used to exploit demonstrations to generate on-line a suitable movements for a large range of situations.

As can easily be imagined, generally speaking employing the feedback improves the performance over open-loop [19] playback. In short, with open-loop the only information available when taking actions is the internal representation of time. Conversely, using sensory inputs the control algorithm can have a more complete knowledge of the current state and can take appropriate actions to achieve the desired robot state.

This anyway requires a design (or at least tuning) of the relations between the sensory inputs and the control action taken. The approaches taken are often both strongly task and robot dependent. Our aim is to devise a technique that exploits the information coming from the sensors to improve the system robustness. Obviously any comparison with specifically designed controllers (e.g. ZMP based) will be omitted, since we can easily guess that the task and robot specific knowledge exploited in the design phase can strongly improve the performance of those algorithms compared to our completely general technique. To be more precise, our goal is to design a simple, task independent technique that

corrects deviations from the expected sensory inputs of essentially stable systems due, for instance, to environmental changes. Drastic changes of the movement, inverse dynamics computation or complete replannings such as using the hands for support during walking [20] are outside of our scope. We therefore assume to have perturbations in a range such that trying to track the sensory states is sufficient to stabilize the motion.

Specifically, we assume the robot to have the body covered by force sensors, as in [21] and the actuators equipped with potentiometers/encoders. Our approach consists in two phases. In a first, off-line phase we play the motion in open-loop in a reference environment. We record the sensory information and store the transitions between sensory states as a graph. This graph becomes our motion representation. In a second, online phase, we use the information contained in the graph to replay the motion.

This is very similar to the StateNet approach [22] and presents the same advantages (quantitative comparability between the states, easily extendable robot's knowledge representation, online search of correction strategies). However while in [22] a transition is a complete movement like "Stand up" our edges represent small state transitions and so a motion like standing up can be decomposed into dozens of states. This difference in granularity reflects our aim to make small corrections to cope with environment disturbances. In fact [22] deals with complete behaviors and, for instance, if the robot falls down while walking then it will initiate a roll over, sit and stand up sequence. In our case we want to restrict the recovery actions to small modifications of the motion to respond to disturbances. The higher granularity of the state transitions allows us to utilize noise to recover from stuck conditions. For instance, if the robot gets stranded while crawling on a rough terrain, making random perturbations of the movements can help in escaping from the locking condition. While with the StateNet framework a completely different behavior (for instance, crawl backwards) could be instantiated our approach will tend to generate a movement very close to the original one. Drastic changes in the behavior to escape from really critical conditions could be introduced in our approach by an higher level decision level with the role of switching the target behavior from crawling to, for instance, sidestepping.

In section II we provide the details of the graph construction and exploitation. Section III reports results obtained by experiments with a simulated robot. In detail the robustness of the motion representation and the possibility to employ noise to recover from stuck condition is tested simulating the execution of a crawling motions under different conditions. Simulation results show impressive improvements in comparison to a classic open-loop feedforward playback. We conclude in section IV by illustrating future works.

II. Graph representation

The idea of storing multiple execution of a motion as a set of states is not new, and we can find examples both for the Computer Graphics field [23], [24] and in the description of robot behaviors [22]. Assuming we have a precomputed motor command sequence we play it open-loop, sample the sensory information and construct the graph. Specifically, each node represents a hypercube in the sensory space, since we assume two sensory states to be mapped to the same node if their infinity norm[1] distance does not exceed a threshold Δ (in other terms a node corresponds to a point in the sensory space with a confidence interval of size Δ). Each edge represents a motor command that causes a transition between states. Every edge is labeled by a weight that indicates the number of times that the transition was executed in the off line phase. In the online phase at each sampling time we identify the node corresponding to the sensory information. If the state belongs to one of the nodes identified in the off line phase we just execute the transition with the highest weight. If the node is unknown we determine the nearest node explored in the off line phase and issue a command that should bring the robot to that state. Note that since we simply compare the postures of the two states and avoid any dynamics consideration, the system is not guaranteed to move to the desired state. As done in [22] we assume that moving between close states has a strong success likelihood. In case the resulting state is different from the planned one, the system will simply search the closest known state in the next step too.

This is an important constraint in the choice of the sensory information contained in the nodes. In fact we need to be able to estimate, on-line, a command able to bring the robot to the desired state (nearest known node hypercube center). One possible choice, which is what we actually did, is to include proprioception information (state of the actuators, e.g. joint angles) in the sensory state, so that the motor command to be issued to reach the desired state can be obtained by a transition from the current joint angles to the desired joint angles.

A. Graph construction

Let us denote the nodes and the edges of the graph $G = \{N, E\}$ respectively by $N = n_1, \ldots, n_P$ and $E = e_1, \ldots, n_Q$ where P is the number of nodes and Q the number of edges. Let us then indicate using s_i the sensory information corresponding to the center of the hypercube defined by the node n_i and by w_j the weight of edge e_j. Initially the graphs consists of a single node, n_0 corresponding to the robot's initial state s_0 with no edges. The previous node n_* is initialized to n_0. At each sampling instant we apply the following algorithm

Read the sensory information $\hat{s}$
if $\|s_* - \hat{s}\|_\infty > \Delta$ **then**
  if n_k such that $\|s_k - \hat{s}\|_\infty \leq \Delta$ exists **then**
    if edge e_j between n_* and n_k **then**
      $w_j \leftarrow w_j + 1$
    else

[1]We chose to employ the infinity norm for its low computational cost. Any other norm should also be valid.

Add edge e_{Q+1} between n_* and n_k
$w_{Q+1} \leftarrow 1$
end if
$n_* \leftarrow n_k$
else
Add node n_{P+1}, a hypercube centered at $\hat{s}$
Add edge e_{Q+1} between n_* and n_{P+1}
$w_{Q+1} \leftarrow 1$
$n_* \leftarrow n_{P+1}$
end if
end if

Note that self loops are avoided by the first control of the algorithm. In our implementation the motor command information associated to edge e_j from node n_A to node n_B include a histogram of the time spent on node n_A before taking the transition between node n_A and node n_B. This allows the representation of motions where the robot keeps still for some time which would otherwise not be possible. For simplicity we assume the sampling frequency to be high and the value of Δ to be small enough so that motor commands can be approximated by linear transitions between the angles stored in the nodes at the head and the tail of the edges. Under these simplifications the motor command associated with each edge can be simply derived by the angle difference between the centers of the hypercubes of the head and tail nodes. To avoid perceptual aliasing [25] occurring if we had motions that contain repeated sub-movements, we included time as state information stored in the nodes (note that for periodic motions the time returns to 0 at every cycle). Summarizing, the motion description we obtain from the off-line phase is a graph where

- Nodes represent the robot state. Each node contains information on
  - joint angles (as measured by the simulated encoders)
  - forces acquired by the tactile sensor (simulated FSR that provide the perpendicular force applied to each surface)
  - time instant of the motion execution
- Edges represent state transitions. Each edge contains information on
  - the number of times the transition was taken (stored as the node weight)
  - a histogram of the time taken to perform the transition

Given the heterogeneity of the units of the node, these are normalized by normalizing factors. In detail, joint angles and forces are divided by 3 times the standard deviation of the sensor errors and the time instant by 50 ms (assuming this is a lower bound under which no perceptual aliasing occur).

B. Motion reproduction

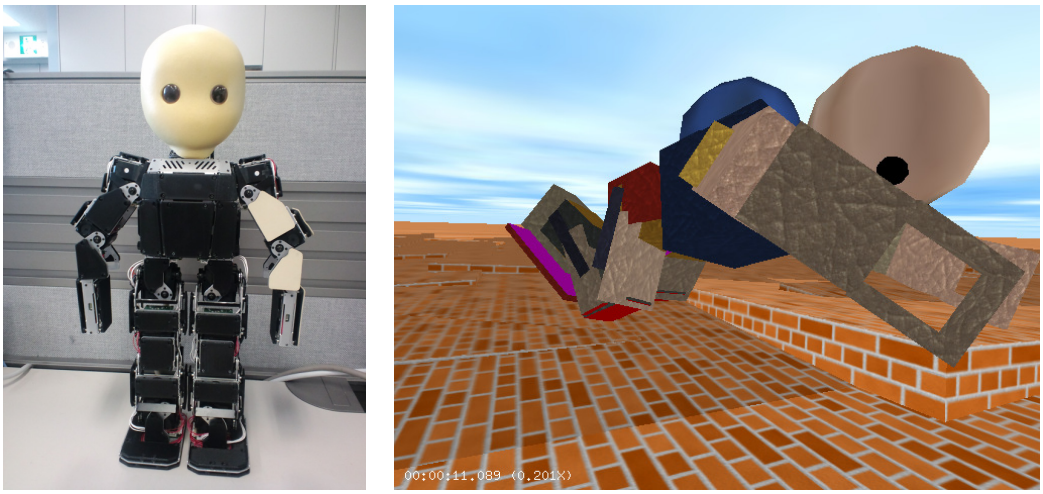
The graph obtained by the off-line phase constitutes a robust motion representation that contains in a unified framework sensory information together with motor commands. During the motion execution, a "virtual time" is used to avoid forcing the motion evolution to have the same period as the trajectories used to generate the graph. To state it more exactly when a motor command is issued the current time is assumed to be the time of the target node, whether the target state was actually reached or not. This virtual time is also used to avoid loops: assume the robot is in state A and tries to reach state B, but ends up in state C. Imagine then that B is the closest known node from C, and that trying to reach B makes the robot remain in state C. Forcing the algorithm to choose the closest node with a time higher than the virtual time is sufficient to avoid the robot to get stuck in such loops. The simple algorithm used for motion reproduction at each sensor sampling time is the following. t^* denotes the virtual time, while n_i^t indicates the time of node n_i. To allow periodic motions, the transitions are accepted also toward nodes with a highly smaller time, so that the motion can "restart" (case *or* $n_i^t - t^* < -\Theta$ in the algorithm). The value Θ is set to the 90% of the period in our implementation.

Read the sensory information $\hat{s}$
$c \leftarrow \arg_i \min \|s_i - \hat{s}\|_\infty$
if $\|s_c - \hat{s}\|_\infty \leq \Delta$ **then**
$h \leftarrow \arg_j \max\{w_j : e_j \; departs \; from \; n_c\}$
execute the motor command e_h
$t^* \leftarrow n_b^t$ where n_b is the node pointed by e_h
else
$d \leftarrow \arg_i \min \|s_i - \hat{s}\|_\infty \; s.t. \; n_i^t > t^* \; or \; n_i^t - t^* < -\Theta$
issue a motor command to reach state s_d
$t^* \leftarrow n_d^t$
end if

As described in the previous subsection, in our implementation each edge memorizes a histogram of the time spent to execute the state transition. When edge e_j is chosen the reference angles sent to the motor's PID controllers vary (linearly) from the position specified in the tail node to the position specified in the head node. The time over which this variation is performed is equal to the mode of the time statistics store in e_j.

When the system is in a state not belonging to the nodes generated during learning (else branch in the previous algorithm), a motor command corresponding to a variation between the current angles and the angles specified by the desired state s_c is issued. In this case the command spans over a constant time T_C (50 ms in the current implementation).

We can soon notice that it is be possible to prune the graph before motion replaying, leaving a single outgoing edge from every node (the one with the highest weight) and storing just the mode instead of the complete waiting time statistics. In this case, the complexity of the algorithm becomes $\mathcal{O}(P)$, i.e. linear in the number of nodes since at each sampling time we need to search the closest node. In our experiments nodes were always in the order of few hundreds, so the computation is feasible with low computational power. If P increases and the computation becomes too slow the Best Bin First algorithm [26] can be employed.



(a) (b)

Fig. 1. VisiON 4G equipped with touch sensors and its simulation on rough terrain.

III. Experiment

The described algorithm has been tested by obtaining a graph representation of a crawling motion performed by a humanoid robot. The experiment was conducted by employing a simulator based on ODE[2], that models the VisiON 4G, a small 22-degrees of freedom humanoid robot produced by VStone and equipped with touch sensors all over the body, as visible in Fig. 1.

Specifically we developed a motion from scratch using a frame based motion editor (i.e. the robot motion is defined as a set of key-frames, that is, a set of instants in time for which the position of each and every joint is provided). We executed the motion 100 times on a flat terrain and constructed a graph with the algorithm described in subsection II-A. Even if the simulator does not include an explicit generation of sensor noise, numerical errors and damping of the lateral oscillation generated by the first step prevents the various executions from being identical. The resulting graph, obtained for $\Delta = 2$ consisting of 66 nodes and 158 edges is reported in Fig. 2. We then conducted three experiments

1) We verified whether the graph representation is able to store the original, trajectory based representation
2) We verified the robustness of the approach under several environmental changes
3) We added noise to the servomotor signals, in order to get the robot unstuck on very rough terrains, and verify that also in this case we can take advantage from the graph representation over the feed-forward solution

We defined the crawling motion performance as the average velocity of the robot. Expressly, since during the execution of the crawling motion lateral swings are present and therefore instantaneous velocity of the center of mass is not very meaningful, we defined the velocity at time t as

$$v(t) = \frac{\left\| \int_{\tau=0}^{T} X(t-\tau)d\tau - \int_{\tau=T}^{2T} X(t-\tau)d\tau \right\|}{T}$$

where X(t) is the (3D) position of the robot's center of mass at time t, T is a constant (4 seconds) and Euclidean norm is used.

[2]http://www.ode.org

A. Motion replay

We verified practically that the graph representation is sufficient to store the crawling motion, as well as other motions like sitting down, turning over, crawl turning left and right. In particular, unexpectedly, we could see that using the graph representation strongly improves the performance even if the environment used to replay the motion is identical to the one used to construct the graph: the average velocity (measured over 4 minutes) is 5.25 cm/s for the open loop execution and 7.86 cm/s for the graph based approach (150%). This is quite surprising if we consider that the motions was hand tuned for this environment.

The performance improvement is due essentially to two factors. First, oscillations are quickly damped using the graph, in fact if we calculate the variance of the roll angle during the motions we get 0.0084 rad^2 for the open-loop case and 0.0047 rad^2 for the graph based control. Secondly, the average crawling period is reduced. Both these effects are due to "shortcuts" on the graph, which are performed when the robot reaches unexpected states, that make the robot skip small parts of the graph. Figure 4(a) shows the nodes and paths visited in the online phase. Figure 3 shows part of the graph, relative to the transition between states 53 and 5. For instance the open-loop control visited states 53, 0, 54, 0, 1, 54, 55, 3, 63, 64, 63, 3, 2, 3, 4 and 5. The short loops that start and end in states 0 and 3 lasted, respectively 6 and 39 milliseconds, and the complete transition between states 53 and 5 took 297 ms. Using the graph the robot arrived at state 53 and moved to state 54 (reached after 46 ms), planned to go to state 0 but ended up in state 4 (at time 96 ms) and then proceeded to state 5 (reached 158 ms after the reaching of node 53). In this case, therefore, the small loop among nodes 3 63 and 64, that consists of nodes where the robot doesn't

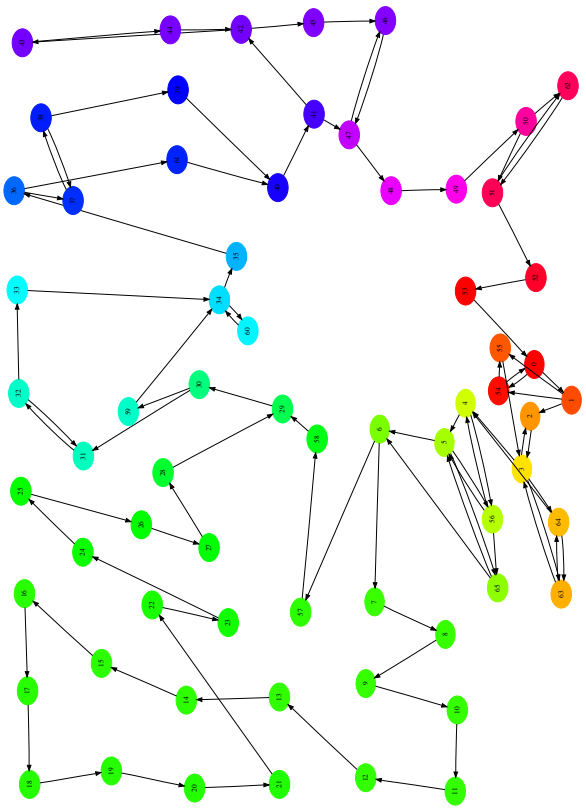


Fig. 2. Graph representation obtained for the crawling motion. The color (hue) indicates the execution time (of the feed forward motion) for which the node were created using the HSV scale.

move the motors but oscillate laterally due to the dynamics, was skipped. This didn't affect the motion, but, conversely, improved the performance. Note that the inclusion of time in the nodes to prevent state aliasing also prevents clamping big loops that describe meaningful parts of the movement (for example one complete step) since no "shortcuts" will be created between nodes that have similar touch sensor and angle information but very different times. However, future works will investigate whether the assumption that "small" loops can be neglected is true, and will possibly introduce on-line learning that automatically identifies the necessary loops.

B. Motion replay under different conditions

To verify the robustness of the approach we compared the performance of the open loop motion replay and the graph based execution under several environmental changes:

- Reduction of the friction coefficient with the floor to the 10% of the reference value: open loop 4.84 cm/s, graph based 8.12 cm/s (168%)
- Halve the robot's weight: open loop 4.62 cm/s, graph based 7.22 cm/s (156%)
- Walk on an ascending slope of 20 degrees: open loop 2.81 cm/s, graph based 5.79 cm/s (206%)
- Walk on rough terrain: open loop 0.54 cm/s, graph based 1.37 cm/s (249%)

Where the rough terrain (depicted in Fig.1) is obtained in the following way:

1) construct a square grid of 50 cm interspaced points on the floor
2) place a tile of 70*70*14 cm centered in each of the grid points
3) apply a random translation on the X-Y (floor) plane of each tile (uniformly distributed in the range $[0, 30]$ cm)
4) change the pitch and roll of each tile by a random angle (uniformly distributed in the range $[0, 3]$ degrees)

C. Addition of noise to avoid stuck conditions

The performances on very rough terrains like the one described in the previous section can be strongly improved if noise is added to the servomotor commands when the robots gets stuck. We decided to add Gaussian noise with standard deviation that is a decreasing function of velocity. This is very similar to what happens in biological systems: the actions taken are more and more deterministic the better the conditions are and the more and the more stochastic the worse the conditions are. For instance, E. Coli proceeds alternating forward movements and random tumbles, and decreases the frequency of random tumbles when the environment conditions are favorable [27], [28] (i.e. positive gradients of nourishing chemicals). In detail we decided to use a simple piecewise linear function to map velocities to the noise standard deviation

$$\eta(v) = \begin{cases} \alpha & if\ v \leq v_\alpha \\ \alpha + \frac{v - v_\alpha}{v_\beta - v_\alpha} * (\beta - \alpha) & if\ v_\alpha < v \leq v_\beta \\ \beta + \frac{v - v_\beta}{v_\beta - v_\gamma} * (\gamma - \beta) & if\ v_\beta < v \leq v_\gamma \\ \gamma & if v > v_\gamma \end{cases}$$

We determined a good set of values for this piecewise function, namely $\alpha = 0.035$, $\beta = 0.014$, $\gamma = 0.0064$, $v_\alpha = 3.8$, $v_\beta = 5.9$ and $v_\gamma = 6.4$ using a genetic algorithm (population size 20, 50 generations). Units are cm/s for velocities and radians for the standard deviation (we apply noise to the motor target angle). Before executing each motor command the robot's velocity v_t is evaluated (with the definition of instantaneous velocity previously provided) and the target angles are perturbed with Gaussian noise of zero average and standard deviation $\eta(v_t)$. To have a fair comparison between the graph based representation and the open loop execution we measured the average state transition time for the graph case and this resulted to be $t_T = 45ms$. We then played the open loop motion approximating it by linear transitions (over time spans of $t_T = 45ms$) between angle postures, where in each interval the target posture is given by the posture specified by the "nominal" trajectory plus a noise given by the $\eta(v_T)$ function. We repeated the experiment for 30 times both for the open loop and graph based representation, obtaining respectively mean velocities of 1.2cm/s (standard deviation 0.94) and 1.9cm/s (standard deviation 0.94) for the two cases.

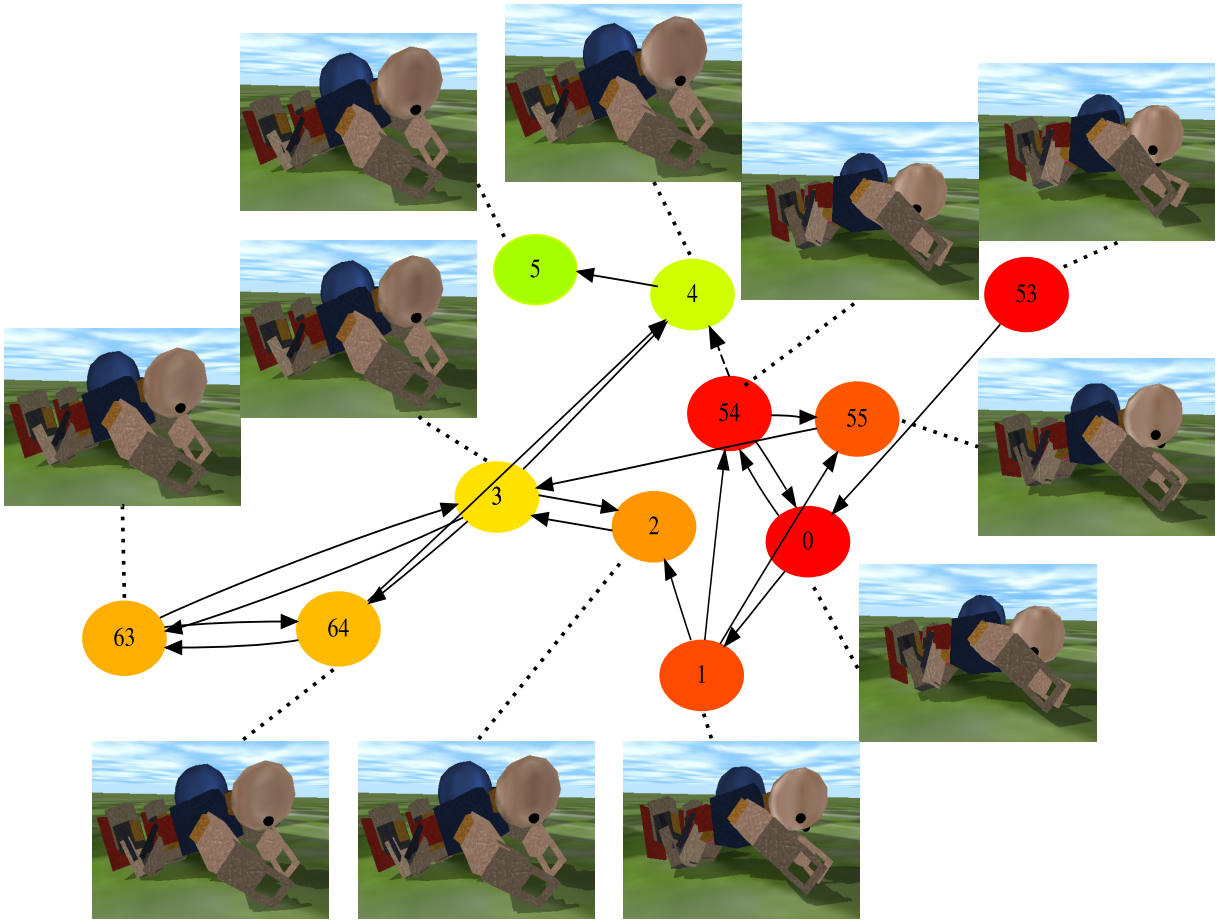


Fig. 3. Part of the graph, with relative postures (each node is linked to a picture of the corresponding state by a dotted line).

IV. Conclusions and Future Works

In this paper we presented a simple technique that introduces sensory feedback in the representation of humanoid robot motions to improve its robustness to environmental changes. Expressly we presented a way to construct a graph of state transitions off-line and how to use this representation to generate motor commands on-line. We performed several simulation experiments on a crawling motion that confirmed this representation to be strongly more robust than a simple feed-forward system. We also showed how simple addition of noise, combined with the graph representation, can help to recover the system from stall conditions. Tests on a real VisiON 4G robot are being performed to test how the noise of real sensors affects our approach.

The presented technique does not require any knowledge of the robot's structure or dynamics and should be completely general. Future works will need to test whether good performances are achieved in other tasks as well, and how much the tuning of the piecewise linear function that adds noise to the system influences the task performances. We do not believe the technique has better performances than specifically designed controllers, e.g. ZMP based stabilizing controllers. Nonetheless given the big difference in the performance with respect to the open-loop system a comparison with such controllers could be interesting.

We also need to investigate the performance in terms of recovering time, i.e. for instance we could compare the time required to bring the robot back to a known crawling state after encountering an obstacle with our and with other classical approaches. We furthermore belive our approach helps robot damage since when unexpected collisions occur the robot will try to bring itself to a known, safe state and not just keep executing the motion in a feed forward manner. Quantitative results will be provided in future works.

A possible improvement of the current system consist of including learning during the motion execution, for instance in order to adapt to a different environment. This could be realized avoiding the graph pruning and updating the graph structure during the on-line phase. However if such modification is included care must be taken not to "forget" the previously acquired knowledge because, for instance, changes on the edge weights could prevent the robot from executing parts of the motion. We will tackle on-line learning (and the corresponding forgetting problem) in future works. Another interesting extension of the work is to use the graph for motion primitives [29] representation. In particular, we can imagine different tasks sharing sub-paths on the graph. Extracting and composing these common subparts could be employed to automatically generate new motions from

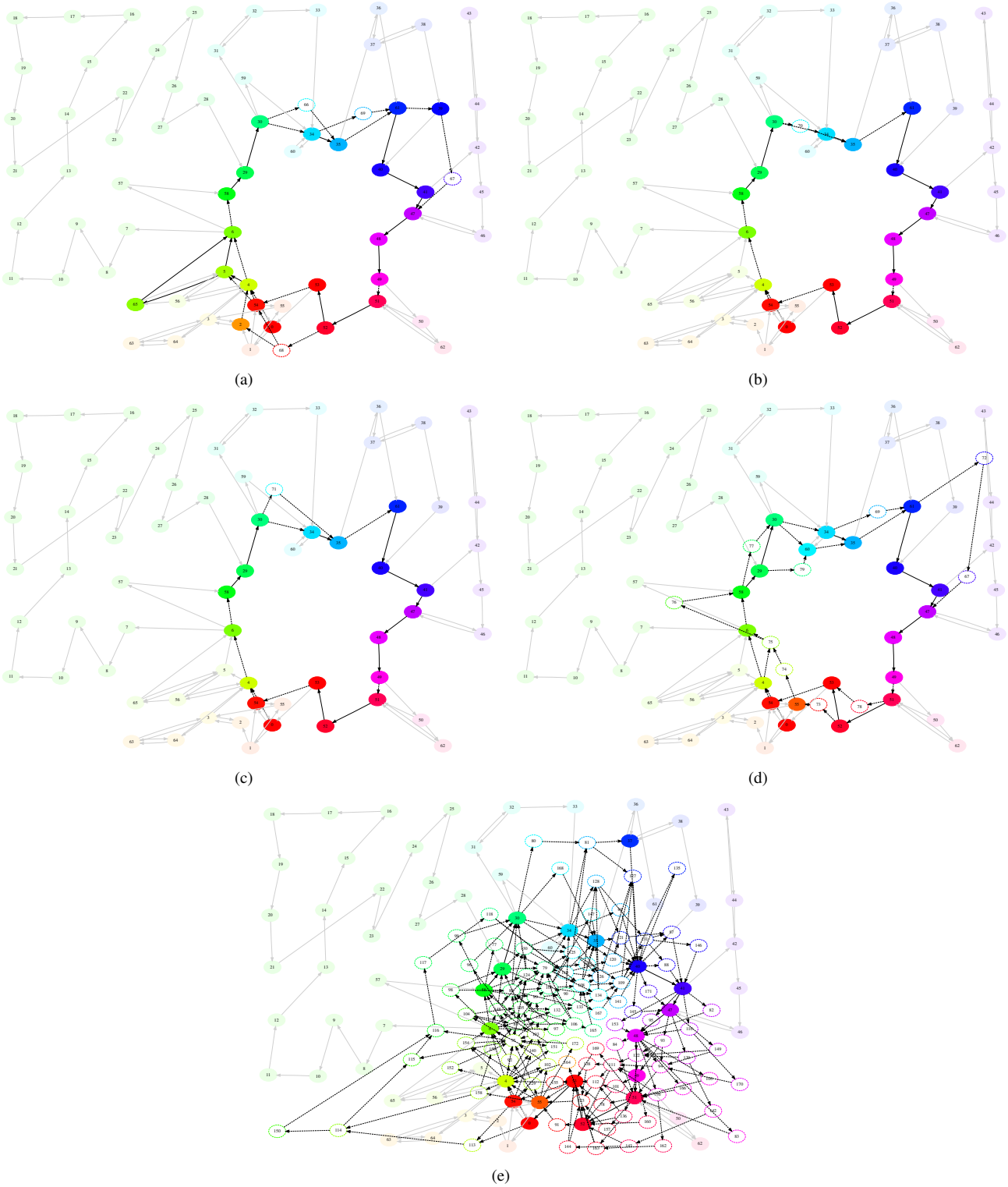


Fig. 4. States visited during the online phase. Filled nodes and solid line edges belong to the knowledge generated offline, while empty nodes are states reached only in the online phase. Color (hue) indicates, using the HSV scale, the creation time of each node. Vivid color (high saturation) for the filled nodes indicates that the node was visit on the online phase, faded color indicates that it was not. Panel (a) shows the data relative to the replay in conditions identical to the offline phase, panel (b) the case of a different friction coefficient, panel (c) the case with the robot's weight halved, panel (d) the case of crawling over a slope and panel (e) the execution of the motion over rough terrain.

scratch,i.e., as done in [22], the graph could be used for motion planning.

V. Acknowledgements

The authors would like to thank Richard Christopher Keely for the comments on the first draft of the paper.

References

[1] K. Harada, M. Morisawa, K. Miura, S. ichiro Nakaoka, K. Fujiwara, and K. K. S. Kajita, "Kinodynamic gait planning for full-body humanoid robots," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS 2008)*, Nice, France, 2008, pp. 1544–1550.

[2] F. Yamasaki, K. Endo, H. Kitano, and M. Asada, "Acquisition of humanoid walking motion using genetic algorithm - considering characteristics of servo modules," in *2002 IEEE International Conference on Robotics and Automation (ICRA 2002)*, Washington, USA, 2002, pp. 3123–3128.

[3] J. Peters, S. Vijayakumar, and S. Schaal, "Reinforcement learning for humanoid robotics," in *IEEE-RAS 3rd International Conference on Humanoid Robots (Humanoids 2003)*, Karlsruhe, Germany, 2003.

[4] S. Iida, S. Kato, K. Kuwayama, T. Kunitachi, M. Kanoh, and H. Itoh, "Humanoid robot control based on reinforcement learning," in *Proceedings of the 2004 International Symposium on Micro-Nanomechatronics and Human Science and The Fourth Symposium Micro-Nanomechatronics for Information-Based Society*, 2004, pp. 353–358.

[5] S. Nakaoka, A. Nakazawa, K. Yokoi, H. Hirukawa, and K. Ikeuchi, "Generating whole body motions for a biped humanoid robot from captured human dances," in *2003 IEEE International Conference on Robotics and Automation (ICRA 2003)*, Taipei, Taiwan, 2003, pp. 3905–3910.

[6] T. Wama, M. Higuchi, H. Sakamoto, and R. Nakatsu, "Realization of tai-chi motion using a humanoid robot," in *IFIP Congress Topical Sessions*, R. Jacquart, Ed. Kluwer, 2004, pp. 59–64.

[7] K. D. Mombaur, H. G. Bock, J. P. Schlder, and R. W. Longman, "Stable walking and running robots without feedback," in *Proceedings of the 7th International Conference CLAWAR 2004*, 2005, pp. 725–735.

[8] Y. Okumura, T. Tawara, K. Endo, T. Furuta, and M. Shimizu, "Realtime zmp compensation for biped walking robot using adaptive inertia force control," in *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS 2003)*, vol. 1, Las Vegas, USA, 2003, pp. 335–339.

[9] S. Kajita, K. Kaneko, M. Morisawa, S. Nakaoka, and H. Hirukawa, "Zmp-based biped running enhanced by toe springs," in *2007 IEEE International Conference on Robotics and Automation (ICRA 2007)*, Roma, Italy, 2007, pp. 3936–3969.

[10] F. Kanehiro, T. Yoshimi, S. Kajita, M. Morisawa, K. Fujiwara, K. Harada, K. Kaneko, H. Hirukawa, and F. Tomita, "Whole body locomotion planning of humanoid robots based on a 3d grid map," in *2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*, Barcelona, Spain, 2005, pp. 1072–1078.

[11] M. Vukobratovic and J.Stepanenko, "On the stability of anthropomorphic systems," *Mathematical Biosciences*, vol. 15, pp. 1–37, 1972.

[12] V. Prahlad, G. Dip, and C. Meng-hwee, "Disturbance rejection by online zmp compensation," *Robotica*, vol. 26, pp. 9–17, 2008.

[13] T. Sugihara, Y. Nakamura, and H. Inoue, "Realtime humanoid motion generation through zmp manipulation based on inverted pendulum control," in *2002 IEEE International Conference on Robotics and Automation (ICRA 2002)*, Washington, USA, 2002, pp. 1404–1409.

[14] K. Yokoi, F. Kanehiro, K. Kaneko, S. Kajita, K. Fujiwara, and H. Hirukawa, "Experimental study of humanoid robot hrp-1s," *The International Journal of Robotics Research 2004*, vol. 23, pp. 351–362, 2004.

[15] H. Inada and K. Ishii, "Behavior generation of bipedal robot using central pattern generator(cpg) (1st report: Cpg parameters searching method by genetic algorithm)," in *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS 2003)*, vol. 3, Las Vegas, USA, 2003, pp. 2179–2184.

[16] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng, "Learning cpg-based biped locomotion with a policy gradient method: Application to a humanoid robot," *International Journal of Robotics Research*, vol. 27, no. 2, pp. 213–228, 2008.

[17] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *NIPS*, S. Becker, S. Thrun, and K. Obermayer, Eds. MIT Press, 2002, pp. 1523–1530.

[18] A. Ude, M. Riley, B. Nemec, A. Kos, T. Asfour, and G. Cheng, "Synthesizing goal-directed actions from a library of example movements," in *IEEE-RAS 7th International Conference on Humanoid Robots (Humanoids 2007)*, Pittsburg, USA, 2007, pp. 115–121.

[19] J. Baltes, J. Anderson, and S. McGrath, "Model-free active balancing for humanoid robots," in *RoboCup 2008 Symposium*, 2008.

[20] K. Hauser, T. Bretl, and J.-C. Latombe, "Non-gaited humanoid locomotion planning," in *IEEE-RAS 5th International Conference on Humanoid Robots (Humanoids 2005)*, Tsukuba, Japan, 2005, pp. 7–12.

[21] T. Yoshikai, M. Hayashi, Y. Ishizaka, T. Sagisaka, and M. Inaba, "Behavior integration for whole-body close interactions by a humanoid with soft sensor flesh," in *IEEE-RAS 7th International Conference on Humanoid Robots (Humanoids 2007)*, Pittsburg, USA, 2007.

[22] F. Kanehiro, M. Inaba, H. Inoue, and S. Hirai1, "Developmental realization of whole-body humanoid behaviors based on statenet architecture containing error recovery functions," in *IEEE-RAS 1st International Conference on Humanoid Robots (Humanoids 2000)*, Cambridge, USA, 2000.

[23] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," *ACM Transactions on Graphics (SIGGRAPH2002)*, vol. 21, no. 3, pp. 473–482, 2002.

[24] H. Sidenbladh, M. J. Black, and L. Sigal, "Implicit probabilistic models of human motion for synthesis and tracking," in *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I*. London, UK: Springer-Verlag, 2002, pp. 784–800.

[25] P. Crook and G. Hayes, "Learning in a state of confusion: Perceptual aliasing in grid world navigation," in *Towards Intelligent Mobile Robots 2003 (TIMR 2003), 4 th British Conference on (Mobile) Robotics*, 2003.

[26] J. Beis and D. G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," in *Conference on Computer Vision and Pattern Recognition, Puerto Rico*, 1997, pp. 1000–1006.

[27] J. Adler, "The sensing of chemicals by bacteria," *Scientific American*, vol. 234, pp. 40–47, 1976.

[28] S. G. Nurzaman, Y. Matsumoto, Y. Nakamura, S. Koizumi, and H. Ishiguro, "Yuragi-based adaptive searching behavior in mobile robot: From bacterial chemotaxis to levy walk," in *2008 IEEE International Conference on Robotics and Biomimetics*, 2008.

[29] O. C. Jenkins and M. J. Mataric, "Deriving action and behavior primitives from human motion data," in *2002 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS 2002)*, Lausanne, Switzerland, 2002, pp. 2551–2556.