A NAO-based Intelligent Robotic System for a Word Search-like Game

Víctor Lobato-Ríos, Angélica Muñoz-Meléndez and José Martínez-Carranza
Computer Science Department
Instituto Nacional de Astrofísica, Óptica y Electrónica, México
Email: vlobato@ccc.inaoep.mx, {munoz, carranza}@inaoep.mx

***Abstract*—In this paper, we introduce a novel application based on the NAO robotic platform and inspired by the *word search puzzle*. In this scenario, NAO is presented with a worktable with letter tokens on it. The goal of this game is for NAO to be given a word that has to be assembled by using the letter tokens on the table. Thus, the robot has to recognise, reach, grasp and bring, to the bottom of the worktable, the tokens following the order of the letters in the word. For NAO to solve this task, we propose a computational strategy based on a vision system for the letter recognition and a motion planning architecture that will enable him to reach and manipulate the tokens. Our results indicate that our approach is adequate and effective to implement this intelligent robotic system, which also provides the basis for the implementation of a more sophisticated robotic system.**

Keywords–Planning; humanoid; grasping; object recognition.

I. INTRODUCTION

Among the several platforms available on the market, the humanoid NAO[1] offers an aesthetic design which makes it appealing among people. Its friendly appearance makes this robot a suitable candidate for companion, however, beyond the cuteness and toy-like appearance, NAO is a genuine robotic platform that can be exploited in several robotic tasks such as object recognition, object manipulation, speech recognition and human-robot interaction.

In this work, we present a novel application of the NAO robotic platform, which contributes to the efforts made to build friendly robots that can sit along humans at home or in social environments for different purposes, *e.g.*, companion, assistance, interaction, etc. In this spirit, we have developed a NAO-based robotic system that aims at solving a game inspired by the *word search puzzle*[2]. In this task, NAO is presented with a worktable where a set of tokens are laid on it, see Fig. 1. Each token has a letter drawn on it and the goal in this game is for NAO to be given a *word* whose compounding letters are found among the tokens.

From the above, in a one-by-one fashion driven by the order of the letters in the requested word, NAO has to: i) recognise, with the help of his vision system, a token whose letter is part of the word; ii) move his left or right hand towards the recognised token on the worktable; iii) grasp the token; iv) move the token towards the bottom part of the worktable and release it in the corresponding position indicated by the order of the letters in the word. The task ends when the word is fully formed as illustrated in Fig. 1, where it can bee seen that, at the bottom of the worktable, NAO has formed the word *CAT*.

[1]NAO robot is developed by the company Aldebaran. For more information consult: https://www.aldebaran.com/en/humanoid-robot/nao-robot

[2]Word search puzzle: A puzzle consisting of letters arranged in a grid, containing several hidden words written in any direction [1].

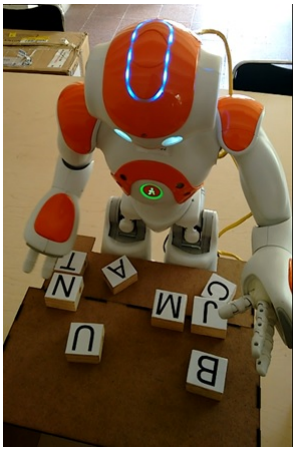


Figure 1. NAO robot assembling the word *CAT*: the tokens with the letters corresponding to the word are placed at the bottom of the table in the respective order.

For NAO to accomplish the task, we have developed a visual recognition system and an efficient planning mechanism that enables NAO to decide which arm to use and the trajectory that it has to follow in order to grasp a token and where to release it. The relevant tokens are recognised with the help of our vision system, which analyses the imagery obtained with NAO's onboard camera. Also, our proposed planner allows us to include relevant constraints in the configuration space such as: possible hand rotations; grasping execution; passing the token from one hand to the other and updates in the configuration space due to accidental (or intentional) changes in the position of the tokens on the worktable.

Our experiments indicate that our approach is effective and the obtained results are promising. They indicate that we are on track in terms of developing an intelligent robotic system that could be used at home or in a social environment in the future.

In order to describe our approach, this paper has been organised as follows: Section II presents the related work; the building blocks of our intelligent robotic system are presented in Section III; Section IV describes our experiments and results; Section V discusses our work and its scope; and Section VI presents our conclusions and future work.

II. RELATED WORK

Several works related to the NAO robot are focused on improving its performance in the RoboCup Standard Platform League[3]. The main topics here are object recognition, color

[3]http://www.tzi.de/spl/bin/view/Website/WebHome

 ISBN: 978-1-61208-437-4

TABLE I. COMPARISON WITH CLOSEST RELATED WORK

	Domain	Resources & capabilities	Real time adaptation
Kovacic et al. [14]	Tic Tac Toe	-Tokens recognition -Tokens' manipulation -Arms motion	No
Jost et al. [13]	Simon's game	-Speech recognition -Arms motion	No
Our work	Word search-like game	-Letters recognition -Tokens' manipulation -Arms motion	Yes

segmentation, gait improvement and fall protection [2][3][4] [5][6]. In this context, many works try to improve specific movements of the robot such as its grasping or its balance while performing a kick, whereas others focus on modelling the kinematics of the robot [7][8].

However, because of its nice appearance, the NAO robot has become a perfect candidate for tasks involving interaction with people, especially in those involving children, elders or people with special needs. For example, Janssen et al. [9] developed a game to motivate children to learn arithmetic through imitation activities between the robot and the child. In [10], NAO is used to learn a set of physical exercises from an expert trainer and then NAO is used to teach the moves to elder people. Additionally, in [11] NAO has been used with the Kinect 3D vision system seeking to imitate upper limb movements of stroke rehabilitation patients.

Further, applications for NAO have also included daily life activities like bringing a cup of coffee to someone [12], where they made the motion plan for each arm in order to accomplish the grasping of a cup and then release it when someone wants to take it.

However, the closest works related to our research have been developed by Jost et al. [13] and Kovacic et al. [14]. The former adapts the Simon's game to a scenario where a NAO robot presents sequences of color that must be repeated and extended by a user. In the latter, a NAO robot plays Tic-Tac-Toe against a person; for that, the robot recognises the game elements visually. Table I compares the main features of these and our work.

III. ASSEMBLING WORDS WITH NAO ROBOT

In order to assemble a word with the NAO robot, which involves letter recognition, grasping and arm motion planning, two main modules were developed. The first module focuses on visual recognition, which analyses the images captured by the onboard camera in order to find the letters, among the tokens on the worktable, that compose the requested word. The second module is responsible for generating the motion plan for the robot's arms with three main goals: i) to move the closest arm towards the token with the recognised letter; ii) to grasp the token; iii) to move the arm whilst holding the token towards the corresponding position at the bottom of the worktable, where the token has to be placed/released. All of the previous steps are necessary in order to assemble the requested word. Both modules and the movement constraints that must be considered will be explained in detail below, but first, a general description of the robot activities will be presented in the platform setup.

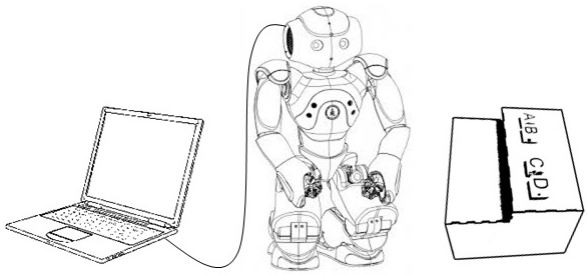
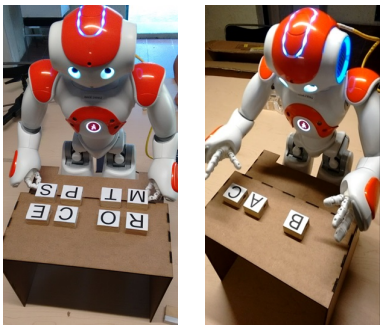


Figure 2. Components of the experimental platform.



(a) (b)

Figure 3. NAO robot with its worktable with: (a) *StandInit* position. (b) Ready position.

A. Platform Setup

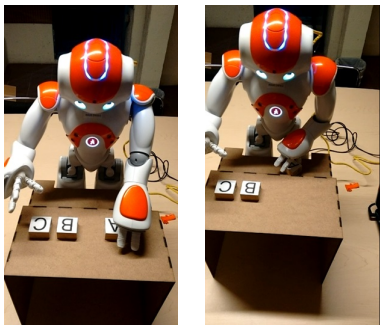
The platform used in this research consists of a NAO robot v4 using the software NAOqi 1.14.4[4]. The robot is connected through an Ethernet cable or through WiFi to a laptop with an Intel Core 2 Duo P8400 processor (2.26 GHz) operating under Windows 8.1. It is worth to remark that the processing of images captured by the robot as well as the planning of arm movements are done on the laptop locally. For a sketch of the platform see Fig. 2.

To begin its activities, the robot must be placed in front of its worktable as depicted in Fig. 3(a). The posture of the NAO is the one defined into the NAO robot environment as *StandInit*, which gives stability and motion freedom to the robot. Immediately after, the robot assumes a little different posture, first, tilts its head to look at the worktable and then raises both arms to chest height as shown in Fig. 3(b).

At this point, the robot is ready to start the visual recognition and obtain the coordinates of the letters that must be reached. As it was stated above, the robot can assemble words by recognising and reaching the letters among the tokens on the worktable as illustrated in Fig. 3(a), where the robot will attempt to assemble the word *MORE*. Furthermore, this same problem is equivalent to assembling a sequence of letters such as it is shown in Fig. 3(b), where NAO will attempt to assemble the letter sequence *A-B-C* from the tokens on the worktable. Either way, the requested word or letter sequence must be previously known by the robot.

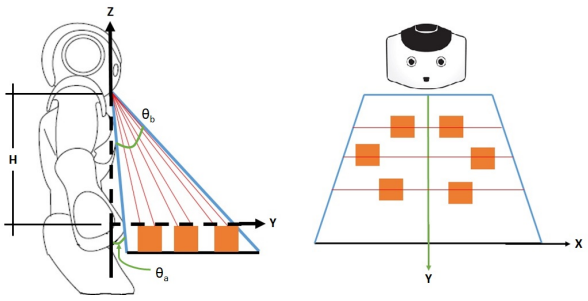
The worktable has two levels, the upper level is where the candidate letters to be recognised are laid on. The lower level is where the robot will bring and release the recognised tokens in order to assemble the word or letter sequence. Once

[4]Documentation of NAOqi 1.14.4 can be found here: http://doc.aldebaran.com/1-14/index.html



(a) (b)

Figure 4. (a) Taking the letter. (b) Releasing the letter.



(a) (b)

Figure 5. (a) Side view of the vision cone of the NAO robot. (b) Top view of the vision cone of the NAO robot.

the coordinates of the letters are known, the robot will grasp these one by one as depicted in Fig. 4(a). NAO will recognise, reach, grasp and bring the tokens by following the order of the letters in the word or in the letter sequence, thus placing these in that order on the lower level (at the bottom) of the worktable as shown in Fig. 4(b).

This procedure ends when the robot has placed every letter of the word or the sequence at the bottom of the table and in the right order. The next section will describe in more detail the vision module, which helps to recognise and locate the letters among the tokens, and how their spatial positions are located.

B. Visual Recognition

Once letters that compose the requested word or sequence are known, the next step is to recognise these letters on the image retrieved by its onboard lower camera. We used this camera because it is located at the mouth of the robot and this makes it easier to see all the surface of the worktable than when using the upper camera located at the forehead of the robot. The captured images have a resolution of 320×260 pixels. However, when using the lower camera, due to its position with respect to the worktable the objects observed with this camera exhibit slight changes in appearance w.r.t. their original appearance, *i.e.*, objects at the bottom of the image are slightly bigger whereas objects at the top are slightly smaller and objects close to the left or right of the image suffer moderate affine transformation. Therefore, the visual recognition model has to be robust against such transformations even if these are small.

To deal with the transformations mentioned above, we use the skeleton of the letter as the template to be sought out on the image where the recognition has to be carried out. To create the template, each token is placed on the worktable in such a way that the letter appears centred on the camera image. Once the letter is well located a segmentation algorithm, based on colour segmentation and morphological operators, is used to extract the white area on the image, corresponding to the token, and then over that area the skeleton is extracted. The resulting template has an average size of 38×42 pixels. Thus, at recognition time, for each letter of the requested word its template is sought out in all the image, the pixel position with the highest similarity score, obtained with normalised crossed correlation, is used as the found image position indicating that the letter has been recognised in that image position.

The procedure described above will return the (x_p, y_p) image position of a recognised letter and in order to obtain the metric token's position (x, y) on the worktable's surface, we use a simple but effective interpolation method that converts a coordinate (x_p, y_p) into (x, y). For the coordinate y, the method uses the angles θ_a and θ_b depicted in Fig 5(a), which correspond to the angles formed in between the camera optical centre and the lower and upper part of the worktable whose sides are observed in the first and last row of the camera image, hence $\theta_y = y_p \frac{\theta_b - \theta_a}{h}$, where h is the total number of lines in the image. From the latter and knowing the camera's height H w.r.t. the worktable, we have that $y = H \tan(\theta_y)$. A similar procedure is carried out in order to calculate x. In this case the vertical length of the worktable is used instead of H, see Fig. 5(a), where the known value of y can be used to simplify the calculations.

C. Movement Constraints

The next module in our work is the motion planning algorithm used to reach and grasp, with the robot's arms, the located recognised letters. This module includes some constraints related to the robot's arm motion, specifically, to its wrist which only has one degree of freedom (DOF) in roll. As a consequence, the robot's hand can pick a token only if the arm and wrist are parallel to the worktable's y axis, see Fig. 5(b). Another constraint is that related to the area that can be reached with either hand. This is mainly due to the DOF of the servo-motors in each shoulder. Fig 6 indicates what tokens on the worktable can be reached with what hand. Due to these constraints and the physical size of the tokens used in this work, the maximum number of letters that the robot is able to assemble is four. Nevertheless, we should highlight that this number could be increased by using smaller tokens or having a humanoid with more DOF but in either case, our proposed methodology would remain the same.

Also, observe that there might be cases when a recognised letter may be located on the left side of the worktable but then, the robot has to place it on the bottom right side. In this case, the corresponding token will be reached and picked by the left hand and then this must be exchanged to the right hand so that the right hand can bring the token to the bottom right side. These steps are depicted in Fig. 7, which demonstrates the ability of our system to deal with these type of situations due to the fact that such cases are considered within our motion planning algorithm.

Finally, we should highlight the fact that our motion

 ISBN: 978-1-61208-437-4

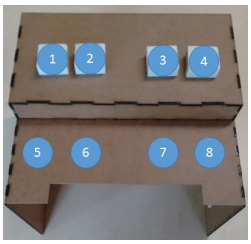
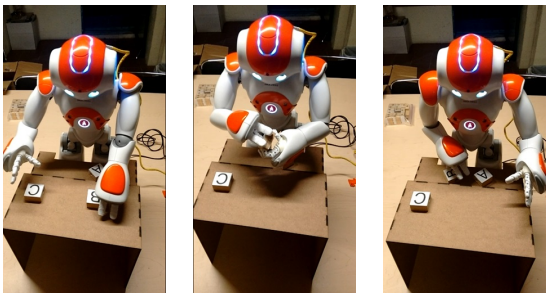


Figure 6. Possible token positions (1-4) and approximate goal positions (5-8).



(a) (b) (c)

Figure 7. (a) Taking the letter. (b) Exchanging the letter. (c) Releasing the letter.

planning algorithm includes in its configuration space the trajectories that the robot's arm has to traverse in order to reach a token. It also includes the situations when either hand has to rotate (usually when there is an exchange of the token from one hand to the other), and the trajectories that will bring a token to its corresponding position at the bottom of the worktable.

D. Motion Planning

As mentioned at the end of the last section, our motion planning algorithm takes into account the arm constraints in combination with the token positions on the worktable, this is, those positions where the tokens have to be reached and picked and those where these have to be released. In order to model all of the arm motions involved during the game, we design a *search tree* which is shown in Fig. 8.

Each node of the tree represents the goal physical positions where the robot's hand has to arrive. This position includes the hand's translation and orientation. In this sense, the labels of nodes indicate the motions for reaching and leaving a token position, represented with the prefixes *R* and *L*, respectively. Also, in order to perform an exchange, nodes are labelled as *REL* to represent the right hand that has to reach the left hand, and as *RER* to represent the left hand that has to reach the right hand. Finally, if an exchange was performed, then the tree includes the nodes with the required motions in order to leave the exchange position, these are labelled with the *LEL* and *LER* in similar fashion to the previous nodes.

Note that the nodes are coloured in either orange or blue, which indicates which hand is manipulating the token. For instance, orange nodes indicate that during the motion execution driven by any of the orange nodes, the left hand could grasp a token or, if it is alreadying hoding it, release it in its corresponding position. The numbers associated with each node correspond to the physical positions of the tokens on the worktable, see Fig. 6.

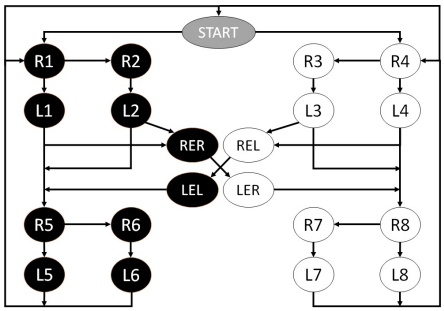


Figure 8. Search tree for planning.

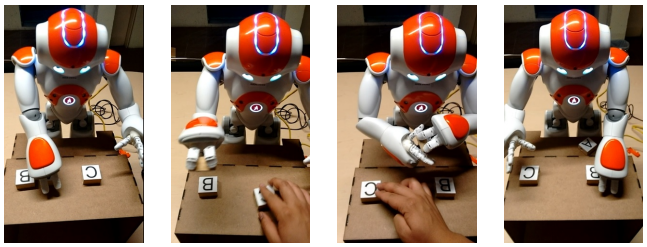
From the above, a node with the label *R4* means that the right arm has to traverse certain trajectory in order to reach the position specified by *R4* and at such position the arm's hand could grasp a token, assuming that a token is physically in that position. If no grasping is required, the arm can move towards another accesible position which is indicated by the arrows in the tree, in this case, from *R4* the arm can move towards *R3*, a token position, and if a token has been grasped then a leaving motion has to be performed in order to bring the token to the bottom of the worktable. The motion planning for this execution will be indicated by traversing the nodes *L3* and any of the other nodes that drive the arm towards a release position, for instance *R8*.

Note that the nodes representing the hand exchanges will contain the required motion for one arm's hand to reach the other arm's hand so that the exchange can be executed. For instance, let's say that the left hand has grasped a token from the position indicated by the node *R2* and it has to be released on the other side of the worktable, then the arm has to traverse the position indicated by the node *L2* and then towards the exchange position indicated by the node *RER*, where the left hand will reach the right hand so the exchange can be executed. Immediately after, the right hand will leave the exchange position following the motion indicated by *LER* and then another node or set of nodes in the tree will have to be traversed to define the motion that the arm will have to follow in order to bring the token to the bottom of the worktable, for instance the sequence *R8-R7*.

In this way, our proposed tree represents all the paths that could be traversed by the robot's arms. Therefore, once a letter has been recognised and its physical position on the worktable calculated by the vision module, the tree will be useful to determine which path is the shortest path in order to drive the arm towards the corresponding token in order to reach it, grasp it and bring it towards its position where it has to be released. This path is found by executing a depth-first search, which produces all the possible paths and that will return the shortest.

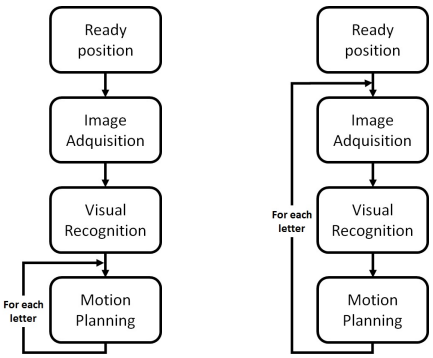
A remarkable feature of the proposed methodology based on our search tree is that once a token has been released in a certain position, defined by a node, for the next recognised letter the search for the shortest path does not have to begin from the *start* robot's position, but it can start from the actual position, *i.e.*, from the current node in the tree. These will save some computational time in the search, but more importantly, it will avoid unnecessary arm motion since having to return to the start position will waste time and energy (NAO's servo motors decrease their performance as time goes by due to

 ISBN: 978-1-61208-437-4



(a) (b) (c) (d)

Figure 9. (a) Taking the first letter. (b-c) Changing the expected intermediate state from *C*-☒-*B* to *B*-☒-*C*. (d) Dealing with changes.



(a) (b)

Figure 10. Computational strategies implemented in this work: (a) Static, the basic strategy used to test our proposed application; (b) Dynamic, which is robust to changes in positions of the tokens during the game.

overheat, hence it is convenient to move the arm as efficiently as possible).

To implement the planner described above, we followed two strategies, see Fig. 10. In a *static strategy*, a picture of the environment is taken only once at the beginning of the game. This image will be used to recognise each one of the letters in the word or letter sequence. However, if the tokens change their position later in time then the planner will not be able to correct the plan since it will believe that the tokens remain always in the same place. The second strategy is the *dynamic strategy* where a picture of the worktable is taken soon after a token has been released, *i.e.*, the planner calculates the position of the next token just before going for it. This will bring robustness against changes in the position of the tokens since it will enable the planner to correctly assign the shortest path given that it will use the updated token's position. Fig. 9 shows an example of the dynamic strategy where a user intentionally changes the position of the tokens while the robot is attempting to bring a grasped token to the position where it has be released. Note that, regardless the change in position, the robot manages to identify correctly the updated positions, hence calculating and performing an adequate plan, something that would fail with the *static strategy*.

IV. Experiments

Three experiments were conducted in order to test our strategies. The first one is a time invested comparison between both strategies in static environments. The second experiment is a time analysis of the dynamic strategy in dynamic environments. Both experiments were performed for assembling sequences of three letters. The third experiment is a test of the performance of the dynamic strategy to achieve the long case.

TABLE II. TIME INVESTED BY STATIC AND DYNAMIC STRATEGIES FOR REACHING THE GOAL SEQUENCE *A-B-C*.

	Average time (seconds)	
Initial state	**Static strategy**	**Dynamic strategy**
A-B-C	76.66 $\pm$ 0.40	76.83 $\pm$ 0.15
B-A-C	92.59 $\pm$ 0.53	92.69 $\pm$ 0.11

The worktable used for these experiments has these dimensions: 30 cm of length, 15 cm of width of the upper level, 12 cm of width of the lower level, 24.5 cm of height of the upper level and 20.5 cm of height of the lower level. Also, wooden tokens of $3.5 \times 4 \times 2$ cm and weight of $11g$ with the letters printed on both sides were used.

All the experiments were run 5 times in similar conditions, and averages and standard deviations were then calculated. It is important to mention that the NAO robot has an automatic monitor for the temperature of its joints. An alert message warns about a situation of high temperature that might cause an unexpected behavior of the robot. Thus, a condition was also that any warning message appeared at the beginning of the experiments.

A movie of these experiments can be watched here [15].

A. Simple Static Environments

The goal of the first experiment was to compare the time invested by both strategies to achieve a sequence of three letters in static environments. The goal sequence was *A-B-C* and two initial positions were tested, *A-B-C* and *B-A-C*. The first one was an easy case because the robot just has to take the tokens and carry them to the goal position in the same order, whereas the second case was harder because it requires two exchanges. The results of this experiment are shown in Table II.

The intuitive hypothesis suggests that the dynamic strategy might be more expensive than the static one since the former invests more time in the process of double-checking the conditions of the environment. However, this hypothesis was not verified by our experiments. Even though the static strategy scored better times that the dynamic strategy, 170 ms and 100 ms for the easy and hard case, respectively, the difference is not statistically significant.

As expected, the harder case takes a longer time, that results from the number of exchanges required to sort the tokens. Note also that the dynamic strategy is in general more consistent in terms of the variation of time invested for solving a problem. That can be explained by the brief pause introduced by the dynamic strategy for the process of double-checking, that benefits at the same time the stability of the robot for starting next movements.

B. Dynamic Environments

The second experiment is intended to verify the ability of the dynamic strategy to react to unexpected changes that might happen in the environment. For that, the goal sequence and the initial states were defined as for the previous experiment, *A-B-C*, *A-B-C* and *B-A-C*, respectively. Once the first token had

 ISBN: 978-1-61208-437-4

TABLE III. TIME INVESTED BY THE DYNAMIC STRATEGY FOR REACHING THE GOAL SEQUENCE *A-B-C* IN CHANGING ENVIRONMENTS.

Initial state	Intermediate state		Average time (seconds)
	Expected	Actual	
A-B-C	☒-B-C	B-C-☒	91.96 ± 0.20
		C-☒-B	85.99 ± 0.14
B-A-C	B-☒-C	B-C-☒	98.78 ± 0.19
		C-☒-B	92.58 ± 0.06

TABLE IV. TIME INVESTED BY THE DYNAMIC STRATEGY FOR REACHING THE GOAL SEQUENCE *A-B-C-D*.

Initial state	Average time (seconds)
A-B-C-D	91.61 ± 0.12
D-C-B-A	123.93 ± 0.07

been taken by the robot, the rest of the tokens were manually resorted in a different configuration. The ability of the dynamic strategy to double-check the state of tokens must be able to deal with these changes. The results of this experiment are shown in Table III.

Two important remarks can be highlighted from these results. First, that the dynamic strategy is effectively able to deal with unexpected changes in the environment, since for all cases including intentional changes of the tokens the robot was able to achieve the goal sequence. And second, that the time invested by this strategy to solve unexpected changes depends more on the specific configuration of the tokens than on the number of exchanges of tokens.

To illustrate the second remark, note that from the initial state, *B-A-C* and the intermediate state *C-☒-B* for reaching the goal sequence *A-B-C* (fourth row of Table III), the robot takes practically the same time invested for going from *B-A-C* with the intermediate state *B-☒-C* for the same goal sequence (second row of Table II). However, solving the same case with the intermediate state *B-C-☒* increases the time in 6 seconds, on average. We have noticed that the time for solving a case is increased particularly for tokens located in middle positions of the worktable, independently of the distance to reach the right position of the token in the goal sequence.

C. Testing with a Large Case

The goal of this experiment was that of testing the performance of the dynamic strategy for solving a larger case, *i.e.*, assembling sequences of four letters that is the maximum number of letters that can be solved by the robot in the current settings.

In this case, the goal sequence was *A-B-C-D* and the initial states were *A-B-C-D* and *D-C-B-A*, an easy and a hard case as for previous experiments. The dynamic strategy was chosen over the static one for its capability to corroborate the conditions of the environment that is convenient when dealing with a large sequence. The results of this experiment are shown in Table IV.

From these results, we confirm that the dynamic strategy is able to solve the largest possible goal sequences in the current settings, as well for easy than for hard cases. Also, it is worth to notice that this strategy scores in general results with small variations, which makes it a stable strategy.

V. Discussion

Programming intelligent robots able to interact with people successfully requires the sum of small efforts in different fields such as improving, for instance, visual recognition algorithms, speech recognition skills, and grasping capabilities. However, a challenging issue that is particularly relevant for this kind of robots is that of how to achieve a flexible goal-oriented behaviour, *i.e.*, how to combine the capabilities for long-term planning with the capabilities for prompt reaction.

A Word Search-like Game using a NAO robot was chosen as a case study for investigating both: issues related to technical aspects of the robot, such as image processing and grasping, and issues related to behavioural aspects of the robot, such as programming strategies for dealing with static and dynamic environments.

The problem itself, a Word Search-like Game, was also defined for a number of reasons that are summarised as follows: i) it is a bound problem, *i.e.*, it has a well-defined set of rules, basic tokens, and initial and target positions; ii) it is a problem whose difficulty can be gradually increased, *i.e.*, variations of the game can be easily extended, for instance by giving NAO spoken commands; iii) it is a problem naturally involving human-robot interaction, *i.e.*, a scenario where a NAO robot plays with children, or assists teachers and helps students to learn spelling is absolutely thinkable.

From the above, a Word Search-like Game using a NAO robot is a problem that contains important ingredients for becoming a solid benchmark for studying human-robot interaction and designing intelligent robots.

VI. Conclusion and Future Work

We have presented a novel application of the NAO robotic platform in the form of a Word Search-like Game. For this, we have developed a strategy that involves two main modules: a vision system that recognises letter tokens on a worktable, and a motion planning module that resolves what motion the robot's arms have to execute to reach and manipulate the tokens in order to solve the game.

Our proposed strategies enable the robot to solve a problem requiring goal-oriented behaviour as well as reactivity against dynamic changes of the tokens' positions. These capabilities are crucial for designing intelligent robots able to interact successfully with people.

Programming automatic players able to successfully play board games and puzzles is a challenging problem in the field of Artificial Intelligence. In effect, there are many non-trivial competences involved in the way people learn and play board games, such as representation of knowledge, identification and refinement of game strategies, and recognition of the opponent's expertise -in the case of interactive games- to mention a few examples.

Programming robotic players able to successfully play board games is a very appealing problem that combines issues encountered in Artificial Intelligence research with physical

 ISBN: 978-1-61208-437-4

and tangible considerations, such as motion constraints, perceptual limitations, time of response, among other.

We strongly believe that the platform and problem addressed in this research are on the track for developing intelligent robotic systems.

In the future, we plan to enhance our approach in order to exploit the walking capabilities of the NAO platform in order to address tasks where the robot has to bring a token to a different location outside the worktable. We will also find potential research scenarios where our NAO-based application can be exploited, for instance educational robotics and robots companions.

ACKNOWLEDGMENT

The first author is supported by the Mexican National Council for Science and Technology, CONACYT, under the grant number 336541. This work was partially supported by the RAFAGA project, funded by the Royal Society-Newton Advanced Fellowship, 2015-2017.

REFERENCES

[1] Oxford Dictionaries, "Definition of: word search". Available on: http://www.oxforddictionaries.com/definition/english/word-search?q=word+search

[2] T. González, "Artificial Vision in the Nao Humanoid Robot", Master's Thesis, Department of Computer Science and Mathematics, Rovira I Virgili University, Sept. 2009, chapter 5-8, pp. 30-77,URL: http://upcommons.upc.edu/pfc/bitstream/2099.1/7722/1/MT_TomasGonzalezSanchez-URV.pdf [accessed: 2015-05-29].

[3] C. Graf, A. Härtl, T. Röfer, and T. Laue, "A Robust Closed-Loop Gait for the Standard Platform League Humanoid", in Proceedings of the 4th Workshop on Humanoid Soccer Robots (Humanoids 2009), Paris, France, 2009, pp. 30-37.

[4] S. Liemhetcharat, B. Coltin, and M. Veloso, "Vision-Based Cognition of a Humanoid Robot in Standard Platform Robot Soccer", in Proceedings of the 5th Workshop on Humanoid Soccer Robots (Humanoids 2010), Nashville, USA, 2010, pp. 47-52.

[5] J. Ruiz-del-Solar, R. Palma, R. Marchant, S. Parra, and P. Zegers , "Learning to fall: Designing low damage fall sequences for humanoid soccer robots", Robotics and Autonomous Systems, vol. 57, Issue 8, 2009, pp. 796–807.

[6] J. Strom, G. Slavov, and E. Chown, "Omnidirectional Walking Using ZMP and Preview Control for the NAO Humanoid Robot", RoboCup 2009: Robot Soccer World Cup XIII. Lecture Notes in Computer Science, vol. 5949, 2010, pp. 378-389.

[7] N. Kofinas, E. Orfanoudakis, and M. Lagoudakis, "Complete Analytical Inverse Kinematics for NAO", in Proceedings of the 13th International Conference on Autonomous Robot Systems (Robotica) Lisbon, Portugal, 2013, pp. 1-6.

[8] H. Mellmann and G. Cotugno, "Dynamic Motion Control: Adaptive Bimanual Grasping for a Humanoid Robot", Fundamenta Informaticae, vol. 112, no. 1, 2011, pp. 89-101.

[9] J. Janssen, C. Wal, M. Neerincx, and R. Looije, "Motivating children to learn arithmetic with an adaptive robot game", Social Robotics. Lecture Notes in Computer Science, vol. 7072, 2011, pp. 153-162.

[10] B. Görer, A. Salah, and H. Akin, "A Robotic Fitness Coach for the Elderly", Ambient Intelligence. Lecture Notes in Computer Science, vol. 8309, 2013, pp. 124-139.

[11] J. Ibarra, A. Malo, A. Gómez, J. Lavín, L. Rodríguez, and W. Sierra, "Development of a system based on 3D vision, interactive virtual environments, ergonometric signals and a humanoid for stroke rehabilitation", Journal of Computer Methods and Programs in Biomedicine, vol. 112, Issue 2, 2013, pp. 239-249.

[12] J. Müller, U. Frese, T. Röfer, R. Gelin, and A. Mazel, "GRASPY – Object Manipulation with NAO", Gearing Up and Accelerating Cross-fertilization between Academic and Industrial Robotics Research in Europe: Springer Tracts in Advanced Robotics, vol. 94, 2014, pp. 177-195.

[13] C. Jost, M. Grandgeorge, B. Le PÈvÈdic, and D. Duhaut, "Robot or Tablet: Users' behaviours on a Memory Game", The 23rd IEEE International Symposium on Robot and Human Interactive Communication, Edinburgh, Scotland, 2014, pp. 1050-1055.

[14] Z. Kovacic, F. Petric, D. Miklic, A. Babic, and K. Hrvatinic, "NAO Plays a Tic-Tac-Toe Game: Intelligent Grasping and Interaction", University of Zagreb, Feb. 2014. Available from https://www.fer.unizg.hr/_download/repository/nao_book.pdf [accessed: 2015-05-25].

[15] V. Lobato-Ríos, A. Muñoz-Meléndez, and J. Martínez-Carranza, "Video: A NAO-based Intelligent Robotic System for a Word Search-like Game". Available on: http://youtu.be/xvMX5glu7Jk

 ISBN: 978-1-61208-437-4