Explicit Knowledge Distribution in an Omnidirectional Distributed Vision System

E. Menegatti †, G. Cicirelli §, C. Simionato †, T. D'Orazio §, E. Pagello †, H. Ishiguro ‡

† Intelligent Autonomous Systems Laboratory
Department of Information Engineering (DEI)
University of Padua, Padova, Italy

§ Institute of Intelligent Systems for Automation
National Research Council, Bari, Italy

‡ Department of Adaptive Machine Systems
Osaka University, Suita, Osaka, 565-0871 Japan

***Abstract*— This paper presents an Omnidirectional Distributed Vision System that learns to navigate a robot in an office-like environment without any knowledge about the calibration of the cameras or the robot control law. The system is composed of several omnidirectional Vision Agents (implemented with an omnidirectional camera and a computer). The first Vision Agent learns to control the robot with $SARSA(\lambda)$ reinforcement learning, using the LEM strategy to speed-up learning. Once the first Vision Agent learnt the correct policy, it transfers its knowledge to the other Vision Agents. The other Vision Agents might have different intrinsic and extrinsic camera parameters (that are unknown), so a certain amount of re-learning is needed. Reinforcement learning is well suited for this. In this paper, we present the structure of the learning system and the discussion about the optimal values for the learning parameters. During the experimentation the learning phase of the first agent has been carried out, then the knowledge propagation and the re-learning stage of three different agents have been tested. The experimental results demonstrate the feasibility of the approach and the possibility to port the system on the actual robot and cameras.**

I. Introduction

Recently Reinforcement Learning (RL) [7] has been receiving great attention by AI and robotics communities [13], [8], [9], [10]. The main attractive of RL is the possibility of a continuous and on-line learning without the need of a teacher which indicates the best association between situations and actions (*policy*). By using RL, the learning agent acquires the control policy by directly interacting with initially unknown environments and then by discovering the best actions in each encountered situation, simply by trying them. The only feedback to the agent, coming from the environment, is the reward (credit or blame) received for each situation/action pair. The final aim of the RL agent is to learn an optimal policy maximizing a value function that estimates the expected cumulative reward in the long term. In this work we apply the RL paradigm to an Omnidirectional Vision Agent which learns to guide a robot from an initial position to a desired one (*target*). Our aim is to build a network of un-calibrated Vision Agents (VA), with overlapping fields of view, able to autonomously learn to navigate a service robot in unmodified human environments. Each VA will take the control of the robot every time it enters its field of view. Several projects explored the possibility to support the human and robot activity in the environment with a network of smart sensors [1], [2]. In [6] we proposed to use a pre-existing network of surveillance cameras to navigate a mobile robot, i.e. to implement an intelligent infrastructure able to support robots' activities in a way similar to the ones proposed in [3], [4]. The difference between the work in [6] and the one proposed in this work is the learning procedure. In [6] neural networks were used to learn to control the robot. However the experimental results showed a poor generalization ability of the neural system due to an overfitting of the training data. In the current work, instead, the application of RL paradigm results to be very appropriate to face the learning problem of each VA.

The aim of this paper is to prove that it is possible to distribute (or "to copy") the knowledge acquired by one VA, in navigating the robot, to another VA different from the previous one and located at a different position in the environment. Notice that each VA can have a different setup of camera, therefore a re-learning phase is needed to adapt the previous knowledge. During learning each VA has to cope with hard problems such as obstacle avoidance and path optimization. The experiments we present in this paper prove the VA is able to learn how to control a mobile robot starting from a zero knowledge about the environment, without the knowledge about its camera parameters and about the robot control law. The acquired policy can be used by new VAs as starting knowledge in order to minimize the learning time and to improve the VAs' performance.

As mentioned above, in our approach the camera of each VA is un-calibrated and the robot does not have any sensor or processing power on board. The cameras are omnidirectional cameras. These type of cameras allow to obtain a complete view of the environment on the contrary of normal pinhole cameras which have a limited field of view.

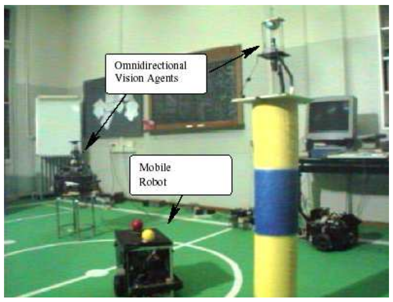


Fig. 1. A picture of the whole system showing two Vision Agents and the robot used in the experiments.

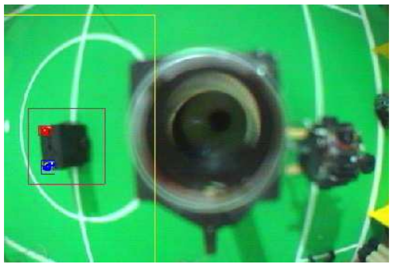


Fig. 2. Sample image taken by the omnidirectional camera.

The robot is just a "dummy" mobile platform which motors are driven by the Vision Agents. The use of uncalibrated cameras is attractive if pre-existing network of surveillance sensors, usually not calibrated, must be used. To calibrate by hand all these cameras can be tedious or even unfeasible, if the number of cameras is large or if the cameras are distributed over a large space. Several researchers are exploring the issue of multiple-camera network calibration, but here we take a different approach: instead of computing the calibration parameters of the camera and using this knowledge to control the robot with a given control law, our system learns at the same time the camera calibration and the control law of the robot.

The system is composed of three omnidirectional cameras with hyperbolic mirrors with a maximum resolution of 640×480 modified to play in RoboCup as a goalkeeper. In this work the omnidirectional camera on board of the robot is removed and the robot is controlled via IEEE802.11 wireless LAN by the Omnidirectional Distributed Vision System. A picture of the whole system showing two VAs is represented in Fig. 1. A zoomed snapshot of the image captured by one VA is shown in Fig. 2.

$SARSA(\lambda)$ with *Replacing Eligibility Traces* has been used as RL method. The choice of $SARSA(\lambda)$ has been done since it is an on-policy method, i.e. the policy is updated by using the approximate values for the current policy. The *LEM* [9] technique has also been applied at the aim of speeding up learning. An optimization analysis of the parameters, used in the applied RL method, has been also carried out at the aim of studying their relevant influence on both the learning time and the success rate. In this first stage of the work, the experimentation has been done in simulation in order to investigate how $SARSA(\lambda)$ method applies to the VA. Moreover, experimentation on the adaptability of the learned knowledge to different VAs has been carried out. The experimental results proved the efficiency and the robustness of the learning system.

The rest of the paper is organized as follows. Next section details how the system has been simulated. Section 3 gives a brief overview of $SARSA(\lambda)$ method. Section 4 defines the state and action spaces. Section 5 and 6 describe the LEM strategy and the study of the principal parameters involved in the implementation of $SARSA(\lambda)$, respectively. Section 7 reports the experimental results obtained by computer simulations. Finally some conclusions end the paper.

II. Simulating the VA

The VA has the task of navigating a mobile robot in its own field of view, choosing the optimal path from the starting position to the target one; choosing the proper velocities to reach the target position fast; avoiding the robot exits from the field of view; avoiding the robot collides with the tripods that support the cameras (*obstacle avoidance problem*); adapting the learned knowledge in case of new situations in the environment. The VA has to carry out all its task without any calibration of the omnidirectional sensor and without any knowledge about the robot control law, but learning by itself the best policy to guide safely the robot (i.e. the best commands to give to the robot for each encountered situation). Since RL paradigm has been considered for learning this policy the necessity of simulating the VA came out. Then we have built a simulator to reproduce all the peculiarities of the VA. The simulator is able to simulate an omnidirectional vision system with different heights of the camera over the floor. The correspondence between the pixel radial distance of a point from the center of the image and the radial distance of the point from the camera in the real world has been experimentally evaluated. Knowing the starting position of the robot the simulator receives as input the pair of velocities $(linear, jog)$ and gives as output the new position of the robot on the image.

III. $SARSA(\lambda)$

In RL problems the learning agent attempts to acquire, by a trial-and-error strategy, the policy which maximizes the expected cumulative reward in the long term. Formally a policy π is a mapping from each state s and action a to the probability of taking action a when the agent is in state s ($\pi : S \times A \rightarrow [0, 1]$, where S is the state space and A is the action space). $SARSA(\lambda)$ is an on-policy control method since it evaluates the policy that is used to make decisions [7]. It differs from an off-policy method which, instead, estimates a policy whereas it uses another policy for the control. The policy π is evaluated estimating the action value function $Q : S \times A \rightarrow \mathbb{R}$ which represents the expected return when the agent performs a given action in a given state.

At each time step, $SARSA(\lambda)$ updates all action values

$Q(s,a)$ according to the following rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))e(s,a) \quad (1)$$

where γ is the discount factor, α is the learning rate parameter, a' is the action the robot takes in the next state s' and $e(s,a)$ is the eligibility trace of action a in state s. In this work we use *replacing eligibility traces* [12] which are updated for all (s,a) as follows:

$$e_t(s,a) = \begin{cases} 1 & \text{if } s = s_t \text{ and } a = a_t \\ \gamma\lambda e_{t-1}(s,a) & \text{otherwise} \end{cases}$$

An eligibility trace is a temporary record of the occurrence of an event, such as the visiting of state or the taking of an action. By using eligibility traces the learning system is able to give out credit or blame to the explored state-action pairs in a more efficient way.

In our experimentation, during the learning phase, the choice among the actions, in each state, is carried out according to the ϵ-$greedy$ policy [7]. It chooses most of the time the actions with the maximal estimated action value, whereas the actions with a lower action value are chosen with probability ϵ. The probability is higher at the beginning of learning whereas it is decreased slowly as learning goes on. This enables a major exploration at the beginning and more exploitation of the acquired knowledge during the advanced learning phase.

Finally the reward function has been defined in the following way: the VA is penalized when it guides the robot outside the field of view of the camera ($r = -60$). On the contrary it receives a positive reward if the robot reaches successfully the target position ($r = 10$). At the other state transitions the reward is evaluated proportionally both to the selected linear velocity and to the angle between the robot heading and the target position. Then the reward is evaluated according to the fact that longer the distance between robot and target position, higher the linear velocity must be. Lower velocities, instead, are preferred as the robot approaches the target position. Similarly the reward is evaluated considering the angle between the robot and the target position.

IV. State Definition

The only source of information the VA has about the environment is the 640×480 image captured from the camera. Figure 3 shows the visible space in an image captured by an omnidirectional vision system. In this image the robot position can be associated to any of about 180600 pixels inside the circular ring with external diameter of $480pixel$ and internal diameter of $20pixel$. The internal black circle contains the self-reflection of the camera. In order to construct a reasonable state space for the VA, we have analyzed its particular task and then defined a state space considering the information really needed by the VA to carry out its task. Then we have identified the following items: the distance between the robot and the target position; the distance between the robot and the camera; the distance between the target position and the camera; the orientation of the robot with respect to the target position; the orientation of the robot with respect to the camera.

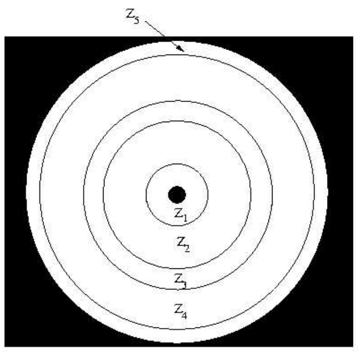


Fig. 3. The five regions of the image space.

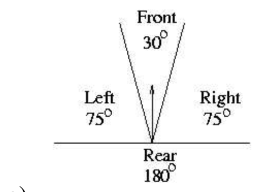
TABLE I
Definition of the classes for the distance d between the robot and the target position

Classes	Distance $d(pixel)$
D_0	$d \leq 4$
D_4	$4 < d \leq 8$
D_8	$8 < d \leq 12$
D_{12}	$12 < d \leq 30$
D_{30}	$30 < d \leq 60$
D_{60}	$60 < d \leq 120$
D_{120}	$120 < d \leq 240$
D_{240}	$240 < d \leq 480$

The distance d between the robot and the target position has been classified into 8 different classes considering a partition of the range interval of d as shown in table I.

Considering the distance between the robot and the camera, the circular ring of the image, described before, has been divided into five concentric regions as shown in fig. 3. The black circle (with $20pixel$ radius) in the center of the image represents the reflexed image of the camera. The black area around the ring is out of the field of view of the camera. Z_1 region has a distance of $30pixel$ from the black circle. Z_1 and Z_5 are alarm regions because of their closeness to the black areas. If the robot is detected in those regions, the VA has to pay more attention to the movement of the robot. Z_5 is $15pixel$ thick. The remaining regions Z_2, Z_3 and Z_4 (with $110pixel$, $145pixel$ and $320pixel$ radius respectively) have been defined considering that a camera with hyperbolic mirror is used and then the resolution of a pixel changes as points approach the center of the image.

Considering the distance between the target position and the camera the ring has been divided only into three regions: $ZT_1 = Z_1 + Z_2$, $ZT_2 = Z_3$ and $ZT_3 = Z_4 + Z_5$ and each of them is classified $free$ or $notfree$ depending on the relative position of the robot with respect to the target position. In particular each region is considered $free$ if the camera does not represent an obstacle for the robot to reach the target position by a straight trajectory. On the



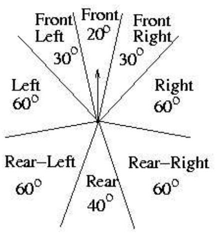


Fig. 4. The angular sectors which define the relative orientation between a) the robot and the camera and between b) the robot and the target position.

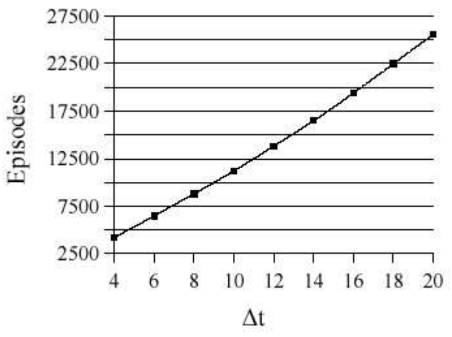


Fig. 5. Learning time versus Δt parameter. Each plotted point refers to the average number of episodes evaluated on 25 simulations.

contrary it is $notfree$ if the the camera is between the robot and the target position.

Finally the relative orientation between the robot and the camera and the one between the robot and the target position have been considered. Fig. 4a) shows the angular sectors (S_C) defined for the camera: $Left$, $Right$, $Front$ and $Rear$. They indicates if the camera is on the left side of the robot, on its right side, in front of it or behind it. For the target position the angular sectors have been augmented in order to optimize the robot behavior. These sectors (S_T) are: $Left$, $Front\text{-}Left$, $Front$, $Front\text{-}Right$, $Right$, $Right\text{-}Rear$, $Rear$ and $Rear\text{-}Left$ as shown in fig. 4b).

Concluding the state of the VA has been defined considering all the elements described above. In particular the state is represented by the 5-tuple (d, Z_i, ZT_j, S_C, S_T) where $i = 1, ..., 5$, $j = 1, 2, 3$. The total number of states is 7680, some are terminal states, many are impossible states. At the end the number of effective states does not exceed 2500.

V. Action Definition

Finally the actions have been defined as pairs of linear and angular velocity $(linear, jog)$ as the robot can receive command in terms of these two velocity values. Each velocity has been divided into sub-actions: 3 for the $linear$ velocity ($stop$, $slow$, $fast$) and 5 for the jog one ($stop$, $slow\text{-}left$, $slow\text{-}right$, $fast\text{-}left$, $fast\text{-}right$). The $(stop, stop)$ action has not been considered as the prime task of the VA is to move the robot in its environment. During the learning phase it may happen that performing one action does not correspond to a state transition (*State-Action Deviation Problem*). To deal with this problem we have adopted the solution proposed in [9]. Each aforesaid action is considered as a *micro-action*. The robot continues to perform one micro-action until a state transition happens. A sequence of micro-actions, named *macro-action*, is considered by the VA as the effective action. In other words the execution of a macro-action produces a state transition then the action value function can be correctly updated.

VI. *LEM* Strategy

In order to improve the learning rate and to speed up learning several strategies have been applied and experimented. In [13], [14] a splitting of the whole task into different parts or behaviors is used. In [15] the learning agent receives advice for figuring out what part of the action space deserves attention for each situation. Asada [9] introduced a new technique known as *LEM* strategy. The learning schedule is constructed such that the agent can learn in easy situations (or missions) at the early stages and in more difficult ones as learning goes on. The difficulty related to the application of *LEM* is the assumption about the knowledge of the order of the states to the target one, but the completeness of knowledge is not needed. In our problem due to the definition of states the agent roughly knows the order of state transitions then a partition of the state space can be applied. We have categorized the state space into sub-sets S_k (missions) by considering first the region where the robot can be, then the orientation of the robot with respect to the target and the orientation of the robot with respect to the camera. So the number of missions obtained is 1120.

Asada proved that in order to shift initial situations into more difficult ones, which means shifting from sub-set S_{k-1} to S_k, the following relation should be satisfied for each $k = 1, 2, 3, \ldots$:

$$\Delta Q_t(S_k, a) = \tag{2}$$
$$= \sum_{s \in S_k} |\max_{a \in A} Q_t(s, a) - \max_{a \in A} Q_{t-\Delta t}(s, a)| < \epsilon$$

where Δt is a time interval which indicates the number of episodes attempted for the S_k mission and should be greater than $|S_k|$, whereas $\epsilon > 0$ is equal to:

$$\epsilon = \alpha \frac{(1-\gamma)}{\gamma} \sum_{s \in S_k} \max_{a \in A} Q(s, a) \tag{3}$$

It is straightforward to prove that the same relations hold also in our case where a different RL algorithm and a different reward function are used. By our knowledge no additional information, instead, can be found in literature about the Δt parameter. We think that it needs particular attention since it is related to the learning time. We have analyzed in greater detail how that parameter affects both the learning time and the agent performance.

Figures 5 and 6 show the results obtained after running a number of simulations for different values of Δt, averaging

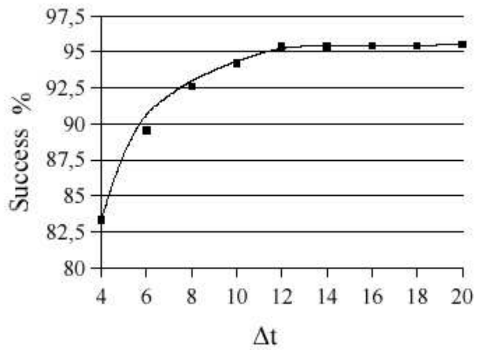


Fig. 6. Percentage of success versus Δt parameter. Each plotted point refers to the average success rate evaluated on 25 simulations.

the result every 25 simulations and with $\alpha = 0.5$, $\gamma = 0.95$, $\lambda = 0.02$ (see next section). Each simulation consists of the completion of a learning phase which starts from a zero knowledge and executes all the 1120 missions defined above. Figure 5 plots the number of episodes versus Δt. As we expected, time grows as Δt increases, but observing the plot we can infer that the relation is practically linear. Figure 6 shows a plot of the success rate (i.e. the percentage of successful episodes) versus Δt. Such a percentage has been evaluated performing 200 test trials after each simulation. Notice that the success rate initially grows but it stabilizes for Δt values higher than 12. This values satisfy the relation $\Delta t > |S_k|$ since in the partition that we have applied to the state space $\max_k |S_k| \simeq 6$, disregarding the unreachable states. In the experimentation we have chosen the value $\Delta t = 14$, because it guarantees a more stable success rate with respect to the value 12.

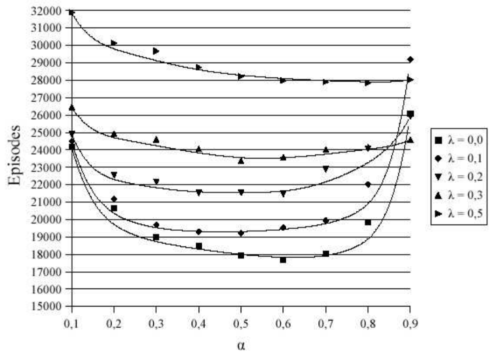


Fig. 7. Learning episodes versus the α parameter for $\gamma = 0.95$ and different values of λ.

VII. Parameter Optimization

The simulated system has been very useful not only to learn and to test the policy of the VA in a virtual environment, but also to study how to choose the best values for the parameters $(\alpha, \gamma, \lambda)$ involved in $SARSA(\lambda)$ method. There's not a standard way to define those parameters to guarantee the convergence of learning. Usually they are heuristically defined and are closely connected with the particular application. In our work we have deeply studied those parameters since they influence the learning time, the success rate and the shifting parameter in the LEM strategy (see previous section). The study has been carried out considering triplet of different values for α, γ and λ, performing 20 simulations for each combination of values and averaging the results. For the evaluation of the success rate 200 test trials have been executed after each simulation.

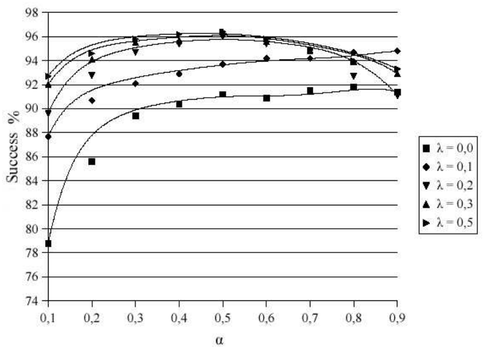


Fig. 8. Percentage of success versus the α parameter for $\gamma = 0.95$ and different values of λ.

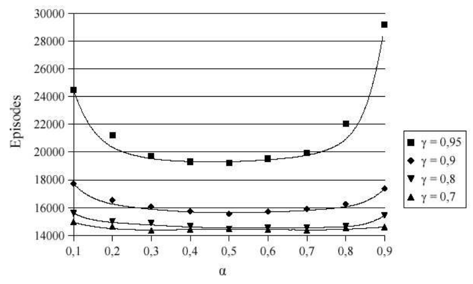


Fig. 9. Learning episodes versus the α parameter for $\lambda = 0.1$ and different values of γ.

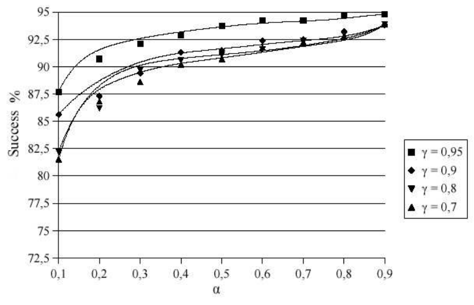


Fig. 10. Percentage of success versus the α parameter for $\lambda = 0.1$ and different values of γ.

Figures 7 and 8 show the results obtained varying α and λ and with a fixed value for γ (=0.95). As can be seen the optimal value for α is close to 0.5 to have both the minimum number of learning episodes and the high percentage of success. As it is expected by applying $SARSA(\lambda)$ the percentage of success grows as λ increases, but also the learning time does the same.

Finally figures 9 and 10 show the results obtained varying α and γ, whereas λ has been fixed to 0.1. As the discount rate γ decreases the number of episodes and also the success rate decrease. This is closely connected to the shifting parameter (see relation (3)) of LEM which is influenced by the discount rate in a deeper way than the learning rate parameter α (see fig. 9).

Concluding, after an analysis of the results, we have chosen the optimal values for α, γ and λ as a trade off between learning time and success rate. The values used in the experimentations are the following: $\alpha = 0.5$, $\gamma = 0.95$, $\lambda = 0.02$.

VIII. Experimental Results

The experimentation has been carried out into two stages: first the VA learns to guide the robot toward the target, then the learned policy is applied to different VAs in order to analyze its adaptability to changes of the environment. The first VA simulates an omnidirectional camera that is $1.8m$ high above the floor.

A number of simulations have been performed to learn the optimal policy. Each simulation performs a learning phase by using the LEM strategy, as described in section VI, and starting from a zero knowledge. During the simulations the VA carries out a number of episodes to complete all the missions of LEM. The average number of episodes performed is 16400. At the end of each simulation the learned knowledge has been tested running 200 additional trial episodes to examine the performance of the VA.

During the testing phase the VA chooses the actions by using a greedy policy and learning is turned off. The average success rate obtained after the testing phase is 95%. Among the different policies obtained after the learning phase we have chosen the best one, i.e. the one with the higher success rate (98.5% in 16100 learning episodes). Figures 11, 12 and 13 show some representative sample paths obtained by using that policy.

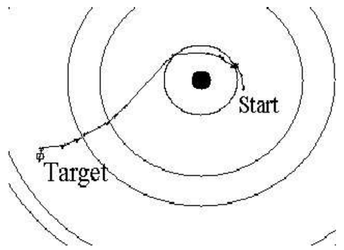


Fig. 11. Sample paths obtained after the learning phase: the robot successfully avoids the obstacle (black circle).

Even though an empty environment has been considered, the camera itself represents an obstacle. Fig. 11 shows that the agent has acquired the capability to avoid the obstacle. The path is drawn with the long vectors which represent the direction of movement, the short vectors, instead, indicate the robot orientation. Observing the figure, it's evident that the VA chooses high velocities at the beginning of the path and lower ones as it approaches the target position. Fig. 12 shows a sample path which underlines this ability. The VA

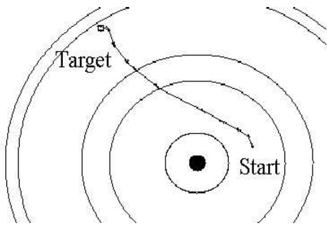


Fig. 12. Sample paths obtained after the learning phase: the robot uses different velocities.

chooses low velocities when the robot moves in proximity of both the obstacle and the target.

Fig. 13 displays the ability of the VA, to keep the robot inside its field of view. In particular the VA moves the robot carefully in the most external region (Z_5) choosing low velocities, then it increases the velocities as the robot moves away from Z_5 and finally it becomes careful again close to the target.

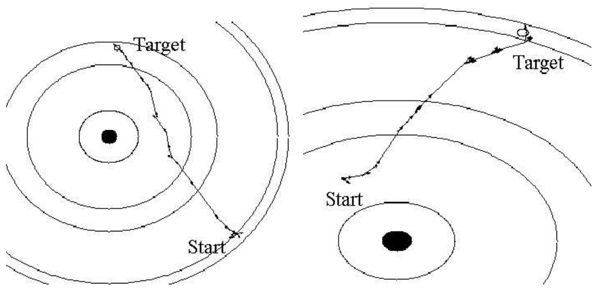


Fig. 13. Sample paths: the VA keeps the robot inside its field of view.

All the presented examples prove that the VA has learned to move the robot by using the proper velocities not only as function of the distance from the target, but also as function of the distance from the camera (calibration ability). In the inner regions the VA has learned to use low velocities even if the target is far from the robot.

Notice that little image distances near the camera correspond to little distances in the real environment, but little image distances far from the camera correspond to large distances in the real world.

In the second part of the experimentation our aim was to prove the distribution of the learnt knowledge to new VAs. In particular two new VAs have been considered in the Distributed Vision System. These VAs have a different camera setup: one has the camera at the height of $2.0m$ above the floor and the second one at $2.4m$. Our aim was to demonstrate that the new VAs can exploit the policy acquired by the initial VA in order to reduce learning time and, more important, to improve their performance. In such a way the same knowledge can be spread over other VAs so that they will be operational from the beginning and they will need only a little amount of re-learning to perform optimally.

The previous learnt knowledge has been used as starting policy for the new VAs, and new learning missions have been defined based on the LEM technique. Partitioning the state space only considering the distance between the robot and the target, we obtain seven missions each one with a cardinality not over 960 states. Several re-learning simulations have been carried out considering different values for the Δt parameter. Recalling that $\Delta t > |S_k|$, Δt has been varied in the range $[1800, 2200]$, obtaining a range of [14000,15400] for the number of learning episodes. Testing the policies on 200 test episodes, after each simulation, both VAs have revealed high success rates ($> 91\%$). As expected higher Δt higher the success rate is. In the next referenced figures the paths executed by using the policies with the highest success rates (97.5% for the first VA and 97% for the second one) will be displayed. Both new VAs are still able to guide the robot toward the target, but a clear improvement of the policy has emerged after the re-learning phase.

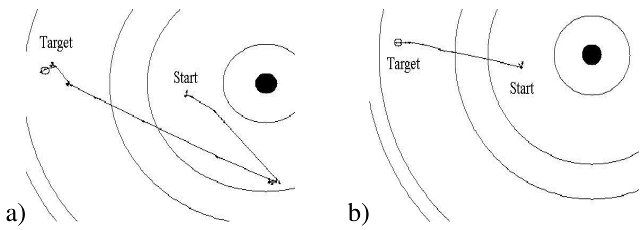


Fig. 14. Sample path a) before the re-learning phase and b) for the VA with $2.0m$ high camera after the re-learning phase.

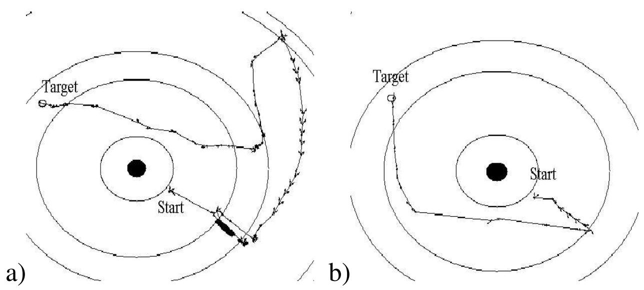


Fig. 15. Sample path a) before the re-learning phase and b) for the VA with $2.4m$ high camera after the re-learning phase.

Figure 14 shows two paths, that start from the same initial position and reach the same target position, before the re-learning phase (fig.14a)) (camera height = $1.8m$) and after the re-learning phase (fig.14b)) (camera height = $2.0m$) respectively. Improved paths have also been obtained by considering the other VA with the camera at the height of $2.4m$ above the floor. Figure 15 shows a sample path for this case. The presented results reveal that the knowledge distribution allows the new VAs to acquire improved control policies. In fact, for the sake of completeness, we have also done another experiment where each VA learnt its own policy starting from a zero knowledge. Apart from the learning time and the success rate, that are comparable with the ones of the previous experiments, what has emerged is that with knowledge distribution each VA exhibits a more optimal behavior.

IX. CONCLUSIONS

This work presents a Omnidirectional Distributed Vision Systems having the task of navigating a mobile robot in its environment. It consists of a number of VAs that autonomously learns to guide the robot from a starting position to a target one. We have applied the RL method of $SARSA(\lambda)$ to learn the control policy. In particular one VA has learnt an initial policy to control the robot, then the same policy has been transferred to other different VAs at the aim of testing the possibility of knowledge distribution. Analyzing the results we have observed that the VAs are able to adapt easily the initial policy to their own peculiarities.

The experimentation has proved that the distribution of knowledge from one VA to other different VAs is possible and that it is advantageous not only for learning time saving, but also for the improvements that the VAs exhibit after the re-learning phase. In fact the VA is immediately operational, because of the starting knowledge, and it learns more and improves with experience since it is allowed to explore again the environment.

Next step of this work will be to continue the experimentation in a real environment exploiting the policy learned in simulation and allowing the real agent to continue its learning in the real situations.

REFERENCES

[1] R. Collins, A. Lipton, and T. Kanade, "A system for video surveillance and monitoring," Robotics Institute at Carnagie Mellon, Tech. Rep., 2000.

[2] H. Takeda, N. Kobayashi, Y. Matsubara, and T. Nishida, "A knowledge-level approach for building human-machine cooperative environment," *Collective Robotics*, vol. 1456, pp. 147–161, 1998.

[3] H. Ishiguro, "Distributed vision system: a perceptual information infrastructure for robot navigation," in *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI'97)*, 1997, pp. 36–43.

[4] H. Ishiguro and M. Trivedi, "Integrating a perceptual information infrastructure with robotic avatars: a framework for tele-existence," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'99)*, October 1999.

[5] B. D. Olsen, "Calibrating a camera network using a domino grid," *Pattern Recognition*, vol. 34, no. 5, 2001.

[6] E. Menegatti, E. Pagello, T. Minato, T. Nakamura, and H. Ishiguro, "Toward knowledge propagation in an omnidirectional distributed vision system," in *Proc. of the 1st Int. Workshop on Advances in Service Robotics (ASER2003)*, March 2003.

[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning: an introduction*. A Bradford Book, 1998.

[8] M. Asada, "A case study on behavior learning for vision-based mobile robot," in *IROS Workshop on Towards Real Autonomy*, 1996, pp. 3–16.

[9] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda, "Purposive behavior acquisition for a real robot by vision-based reinforcement learning," *Machine Learning*, vol. 23, pp. 279–303, 1996.

[10] G. Cicirelli, T. D'Orazio, and A. Distante, "Learning a door-reaching behavior using visual information," in *IASTED-Int. Conference on Control and Applications*, Cancun, Mexico, May 2000.

[11] C. Gomez, Ed., *Engineering and scientific Computing with Scilab*, 1999.

[12] S. P. Singh and R. S. Sutton, "Reinforcement learning with replacing eligibility traces," *Machine Learning*, vol. 22, pp. 123–158, 1996.

[13] S. Mahadevan and J. Connell, "Automatic programming of behavior-based robots using reinforcement learning," *Artificial Intelligence*, vol. 55, no. 2-3, pp. 311–365, 1992.

[14] G. Cicirelli, T. D'Orazio, L. Capozzo, and A. Distante, "Learning elementary behaviors with khepera robot," in *Proc. of first International Khepera Workshop*, Paderborn, Germany, 1999, pp. 109–118.

[15] J. d. R. Millan, "Rapid, safe, and incremental learning of navigation strategies," *Sys. Man and Cybernetics*, vol. 26, no. 3, 1996.