Toward Knowledge Propagation in an Omnidirectional Distributed Vision System

Emanuele Menegatti‡ Enrico Pagello‡ Takashi Minato† Takayuki Nakamura†
Hiroshi Ishiguro†

‡**Intelligent Autonomous Systems Laboratory**
Department of Information Engineering (DEI)
Faculty of Engineering, The University of Padua
Via Gradenigo 6/a, I-35131 Padova, Italy

†*Department of Computer & Communication Sciences*
Wakayama University
930 Sakaedani, Wakayama 640-8510, Japan

Abstract

This paper explores a methodology of knowledge propagation between the Omnidirectional Visual Agents of a Distributed Vision System. The application we chose in this work is to navigate a mobile robot in a real-world environment using a network of uncalibrated cameras. A practical application could be to use an existing camera network intended for surveillance for navigating a service robot in a large scale environment composed of several separated rooms. In this first work, we realised a Vision Agent that learns how to control a mobile robot and propagates this knowledge through the network. The Vision Agent autonomously generates the set of examples used in the learning stage. The knowledge is stored in a neural network, called the **learner**. *Once the Vision Agents has learnt how to control the robot, this knowledge is transmitted to the other Vision Agents in the network. The Vision Agents in the network are heterogeneous, so the knowledge acquired by one agent cannot be used directly by another agent and a certain amount of re-learning is needed. The re-learning process is guided by a second neural net, called* **supervisor**. *We are more interested in the knowledge propagation process than in the knowledge acquisition, we did not focused in the implementation of the neural nets they were used only as a tool. Preliminary results of the experiments are presented.*

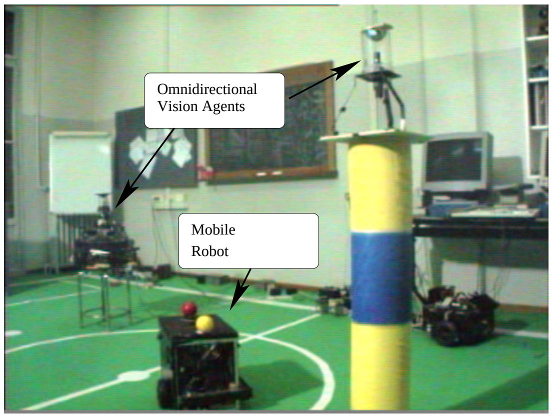

Figure 1: *A picture of the whole system showing two Vision Agents and the robot used in the experiments.*

1 Introduction

There is an increasing evidence that service robots in manned real world environments can take advantages of network of smart sensors. In every day life, we are familiar with networks of cameras spread in the environment (e.g. multi-camera video surveillance systems in banks or airports), but in most cases these are just a set of dummy sensors. They collect a huge amount of videotape data that human operators need to interpret. In the scientific community, there is a strong will to integrate these sensors into an intelligent system. Several researchers are working on autonomous system of surveillance [3] [10] or on intelligent environments [1] [13], like intelligent rooms. In the robotics community as well, the interest of the researchers is shifting form a single agent working in a cell-space to multiple agents working in the real world. For these kind of applications a single camera is not enough and multiple cameras are needed to observe the scene. Again, the aim is to provide an intelligent infrastructure able to support robots' activities [6] [4]. The idea proposed in this paper is that the existing networks of cameras could be used to support the navigation of service robots in the environment. The problem is that usually these cameras are not calibrated. To calibrate by hand all the cameras of a network can be tedious or even unfeasible. Researchers are tackling the issue of multiple-camera calibration. An example is the work of Olsen in which a set of tiles are used to calibrate a large set of cameras observing multiple connected rooms and corridors [11].

In this work we propose a different approach. Instead of having to calibrate all the camera of the network and then to program a robot to navigate in the environment using the information provided by the cameras. We propose a network of uncalibrated vision sensors able to autonomously learn to control a mobile robot in a large scale environment. The cameras in the network can communicate among them and with the mobile robot. To stress the idea that the camera network is more than a simple bunch of sensors scattered in the environment, we called it a **Distributed Vision System** (DVS). The term DVS (Distributed Vision System) emphasises that the camera network acts as a whole, as a single *"super-sensor"*, able to navigate the robot. The DVS is composed of several **Vision Agent** (VA). A VA (Vision Agent) is a vision sensor able to acquire images, to process them and to communicate with other VAs over a network[1]. Every VA is *embodied* in a different sensor and is *situated* at a different position. In our first implementation, the DVS presented in this paper looks like a set of uncalibrated omnidirectional cameras distributed in the environment, see Fig. 1.

In a previous work, we showed how a Distributed Vision System can navigate a robot in a toy-scale real-world environment [4]. That work was quite different from the current work. In that work the learning phase required the intervention of a human operator guiding the robot in the free space in the toy town reproduction. By looking at the trajectory of the robot when driven by the human operator, the Distributed Vision System was able learn the correct trajectory in the free space and then to send information to the robot to autonomously playback the taught trajectory. In this paper, we propose a totally autonomous Distributed Vision System able to learn how to control a mobile robot by autonomously generating a set of examples and by propagating the knowledge through the network. The robot is totally dummy. It has no intelligence on board, it has only

[1]In the taxonomy of Weiss, a Vision Agent can be defined as a *Smart Agent*, because it is autonomous, it is able to learn from experience and it can cooperate with other agents to achieve a global goal [14].

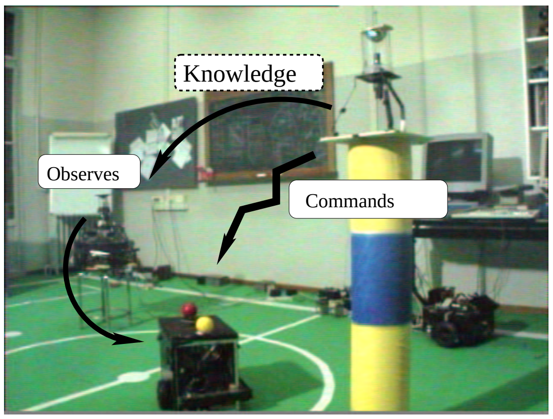

Figure 2: *The knowledge propagation process: the VA on the right send commands to the robot, the VA on the left observing the robot evaluates the knowledge received by the first VA.*

a motor control board fitted with encoders. This make it possible to send to the robot the position to be reached and the motor control board takes care of setting the correct instantaneous translational and rotational speeds to reach the desired position from the current position.

In Section 2, we will discuss in further detail the task of the DVS. In Section 3, we will explain the structure of an Omnidirectional Vision Agents. In Section 4, the learning procedure is discussed and experimental results are presented. Finally, future work are hinted and conclusions are drawn.

2 Learning Process

As we said in the introduction the task of the DVS is to learn to drive a mobile robot. This knowledge is first acquired by a single VA and then is propagated to other VAs in the network. The distributed learning stages is composed of five steps:

1. a Vision Agent learns the mapping between its image space and the robot's motor space;
2. the Vision Agent transmits the learnt mapping to a second Vision Agent;
3. the second Vision Agent checks if the received mapping is valid, if not starts a re-learning phase guided by the first Vision Agent;
4. the knowledge is propagated from the second Vision Agent to the third Vision Agent, and so on;
5. in the end, the sensor network is able to drive the robot in a large environment;

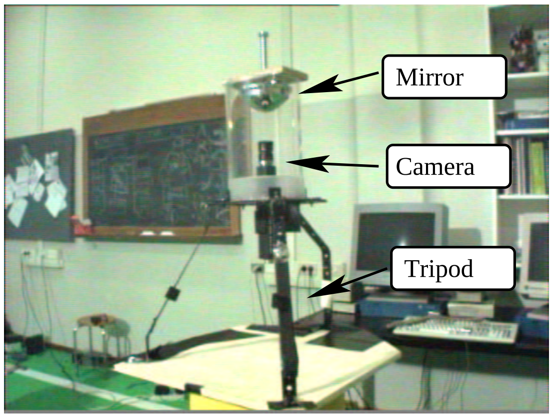

Figure 3: *A picture of the hardware resource of the Vision Agent. On the top the omnidirectional mirror (here a hyperbolic mirror), under the mirror the camera and the tripod supporting the system.*

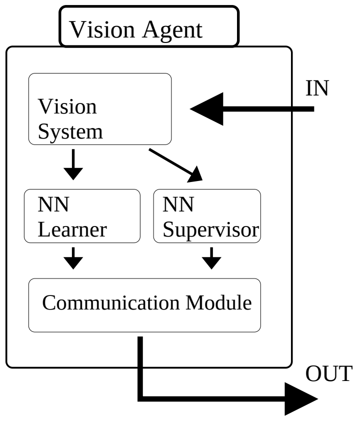

Figure 4: *The simplified scheme of a Vision Agents. The drawing highlights the main blocks composing the VA.*

The bridge to propagate the knowledge among the Vision Agents is the robot. The first Vision Agents controls a robot by sending to the robots a set of random motion commands, see Fig. 2. Looking at the motion of the robot, the system learns the mapping between the position of the robot in the image space and the robot's motor space, see Fig. 5. As we said, the acquired knowledge is then propagated to the other Vision Agents on the network, Fig. 2. Because the vision sensors are not previously calibrated, the propagated knowledge cannot be immediately used and a certain amount of re-learning can be needed. In fact, if the vision sensors are not previously calibrated, the mapping between the image space and the motor space might be different: the cameras could be located at different heights or at different distances from the mirrors, the poles supporting the cameras could be not exactly vertical, etc. In addition, the Vision Agents could mount heterogeneous vision sensors and so the mapping function have to be relearned. Examples of heterogeneous vision sensors can be omnidirectional vision sensors with mirrors with different profiles customised for different tasks [8] [7].

3 Omnidirectional Vision Agent

Every sensor in the network is composed of hardware and software resources. In Fig. 3, the hardware resources of the VA are depicted. The omnidirectional cameras are catadioptric omnidirectional cameras composed of a standard perspective camera and a convex mirror [5]. The camera is pointed upward to the mirror that is reflecting the light gathered by the surroundings of the sensor, see Fig. 3 . Every camera is connected to a PC that provides computational power for image processing and network communication.

The Omnidirectional Vision Agents used in this work are fitted with the following software resources:

- a vision system
- a neural network called *learner*
- a neural network called *supervisor*
- a communication network module

In the next section we will discuss in detail the single modules of the Vision Agent.

3.1 Vision System

The vision system is the sensorial module of the Vision Agent. The Vision Agent gathers information on the world using the vision system.

The task of the vision system is to calculate the position and the heading of the robot in the image. To easily detect the position and the heading of the robot, two coloured balls are fixed on the top of the robot and the vision system looks for these two coloured blobs. The midpoint between the centres of gravity of the two balls gives the position of the robot and the line passing by the centres of gravity of the two balls gives the heading of the robot. In order to avoid to search the whole image for the coloured blobs, the vision system uses a background subtraction algorithm to detect the broad region of image in which is located the robot.

The background subtraction algorithm can be used because we are assuming the Vision Agent is static (i.e. its view of the environment does not change)

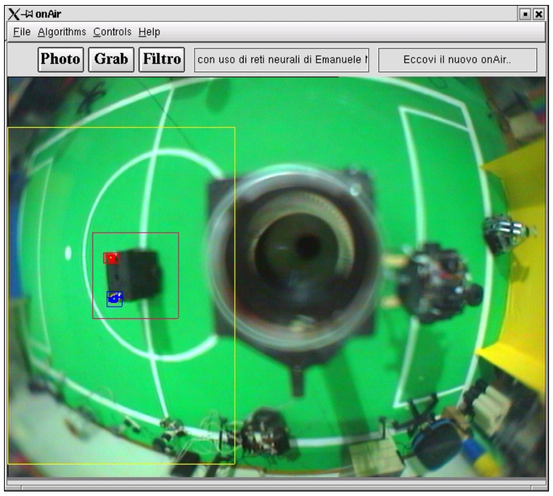
X onAir
File Algorithms Controls Help
Photo
Grab
Filtro
con uso di reti neurali di Emanuele
Eccovi il nuovo onAir..

Figure 5: *A snapshot of the image processing that detects the position and heading of the robot in the image.*

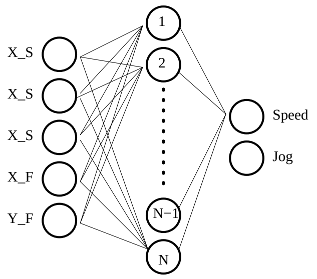

Figure 6: *The structure of the neural network used for the learner network.*

and the robot is the only moving object in the filed of view of the camera.

The final output of the vision system is the heading and the position of the robot calculated both in the polar coordinates centred in the image centre and the Cartesian coordinates in the usual frame of reference for the images.

3.2 Neural Networks

The output of the vision system is passed to the two neural networks. In the learning stage the two neural nets are trained using the robot positions and the robot speeds of the autonomously generated training examples. In the running stage, the task of the *learner network* is to calculate the instantaneous translational and rotational speeds necessary to reach the target position. The task of the *supervisor network* is to check if the commands sent to the robot have been successfully executed.

In this first implementation of our system we used a very simple structure[2] for the neural networks. These are three layer networks with an input layer, an hidden layer and an output layer, see Fig. 6. The units of the networks are sigmoid units and the networks are trained with the back-propagation algorithm [9].

Learner Network. As depicted in Fig. 6, the network has five input units, four hidden units and two output units. The input units take the three parameters of the initial position and heading of the robot and the two parameters of the final position of the robot, respectively named:

$$\begin{aligned} \mathit{input} &\rightarrow \{(x_S, y_S, h_S), (x_F, y_F)\} \\ \mathit{output} &\rightarrow \{(\mathit{Speed}, \mathit{Jog})\} \end{aligned} \tag{1}$$

Note that in the first part of this work we are not considering the final heading of the robot, i.e. the robot has only to reach the final position, but the final heading is not important.

The outputs of the network are the values of linear speed and angular speed to be set on the motors of the robot, respectively.

These values are the instant speeds necessary to move the robot from the starting position to the final position. These values are sent at every cycle of the Vision Agent to the Agent on the robot through the network module.

Supervisor Network. The *supervisor network* takes as input the starting position of the robot and the speeds set on the motors and predicts the final position of the robot. The topology of the Supervisor Network is the same of the Learner Network, but inputs and outputs are respectively:

$$\begin{aligned} \mathit{input} &\rightarrow \{(x_S, y_S, h_S), (\mathit{Speed}, \mathit{Jog})\} \\ \mathit{output} &\rightarrow \{(x_P, y_P)\} \end{aligned} \tag{2}$$

If the final position reached by the robot, as reported from the vision system, is not compatible with the predicted position an exception is raised. This exception means that the learnt control of the robot is no longer valid and a re-learning is needed. As we said, in our experiment this mean that the knowledge received from another Vision Agents cannot be used and a certain amount of re-learning is needed.

3.3 Communication Network Module

The network subsystem used here is a part of the ADE library, a software suite written in C++ and

[2]As we said, we are not interested in the learning process itself, but in the propagation of the acquired knowledge from one Vision Agent to another Vision Agent.

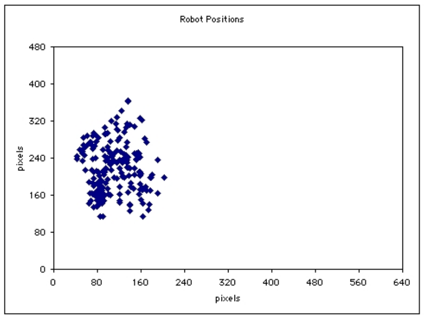

Figure 7: *The positions of the robot for the training data in the omnidirectional image of Fig.5. Every point is both an initial point and a final point.*

used by our RoboCup[3] team: *Artisti Veneti*[4]. ADE provides basic C++ classes implementing thread scheduling, message passing and seamless network communication [2].

The protocol used for data transmission is the connectionless User Datagram Protocol (UDP) over IP. This protocol allows fast network responses even in case of a temporary failure (as it is often the case in noisy wireless networks). Packet loss for our application is not an issue since packets are self-contained and repeatedly sent.

In this application, we send two types of messages: the motor commands to the robot and the messages between the Vision Agents.

4 Training the Neural Networks

The first problem in training a neural net is to have a rich set of unbiased training examples. In the learning phase, the Vision Agent generates a set of random commands for the motors of the robot. The Vision Agent stores in a file the initial pose of the robot, its final pose and the command that moved the robot from the starting position to the finish position. These data are used to train the two networks. In the first experiments we performed the motion of the robot was constrained in a portion of the field of view of the Omnidirectional VA. Fig. 7 shows the position of the robot for each training example. As you can see, the robot moved in the left field of view of the camera, approximately. Every motion had random speed and jog in order to present to the network a set of examples of velocity and jog values not biased by the choices of the user. The linear speed was randomly chosen between -400 mm/s and 400 mm/s.

[3]RoboCup: the Robotics Soccer Championship, URL www.robocup.org

[4]*Artisti Veneti* is the RoboCup team of the University of Padua for the Middle-Size League [12]

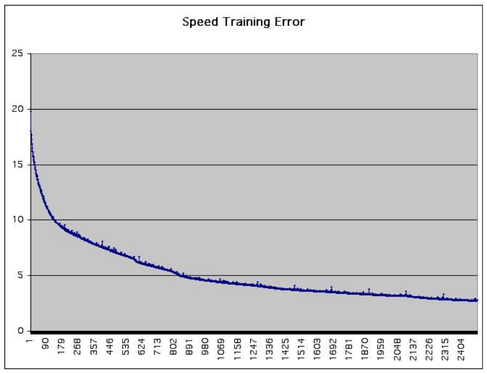

Figure 8: *A plot of the learning error on the speed output versus the number of iterations.*

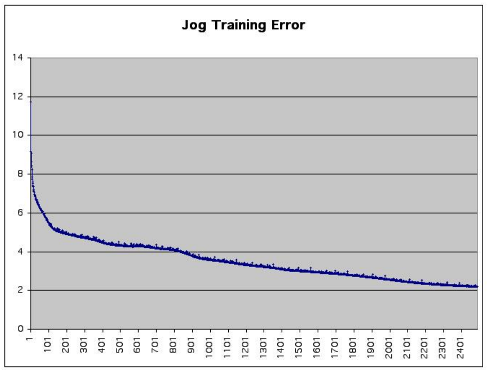

Figure 9: *A plot of the learning error on the jog output versus the number of iterations.*

The angular speed was randomly chosen between -1 rad/s and 1 rad/s. The value of linear speed and angular speed where sent to the robot and the motor where activated with these speeds for 1 second. We collected sets of about 200-300 examples and we used about 160-260 examples as training examples and the rest as validation examples. The training errors for a set of examples is reported in Fig. 8 and in Fig. 9. The two figures respectively refer to the error in the linear speed determination (called *speed*) and in the angular speed (called *jog*).

5 Conclusions

In this paper, we presented a new idea in the creation of an intelligent network of sensor. We proposed a Distributed Vision System that will be able to navigate a mobile robot in a large space. The DVS is composed of several Vision Agents that are able to autonomously learn how to control the robot. This knowledge is acquired locally and is distributed through the network from a Vision Agent to the rest of the Vision Agents using as a bridge the mobile robot for this *knowledge propagation process*. In this paper we focused on the single Vision Agent structure and on the procedure to learn the mapping between the image space and the motor space of the robot. We used a neural network to learn the mapping function. The first experiments show that sometimes the neural network overfits the training data with a lack of generalisation. This results in an error in the validation set that is increasing with the number of iterations in the learning process. At the time of writing we are considering to increase the complexity of the rather basic layout of the network used in this work.

6 Acknowledgements

This research has been partially supported by: the Italian Ministry for the Education and Research (MURST), the Italian National Council of Research (CNR), by the Parallel Computing Project of the Italian Energy Agency (ENEA), by the University of Padua and by the University of Wakayama.

In particular we wish to thanks the students of the University of Padua: Arrigo Zanette, Luca Burrelli and Enrico Ros that helped in the realisation of the experiments.

References

[1] R. A. Brooks. The intelligent room project. In *Proceedings of the Second International Cognitive Technology Conference (CT'97), Aizu, Japan*, August 1997.

[2] L. Burrelli, S. Carpin, F. Garelli, E. Menegatti, and E. Pagello. Ade: a software suite for multithreading and networking. Technical report, Intelligent Autonomous Systems Laboratory, Department of Information Engineering, University of Padova, ITALY, 2002.

[3] R. Collins, A. Lipton, and T. Kanade. A system for video surveillance and monitoring. Technical report, Robotics Institute at Carnagie Mellon University, 2000.

[4] H. Ishiguro. Distributed vision system: A perceptual information infrastructure for robot navigation. In *Proceedings of the Int. Joint Conf. on Artificial Intelligence (IJCAI97)*, pages 36–43, 1997.

[5] H. Ishiguro. Development of low-cost compact omnidirectional vision sensors. In R. Benosman and S. Kang, editors, *Panoramic Vision*, chapter 3, pages pp. 23–38. Springer, 2001.

[6] H. Ishiguro and M. Trivedi. Integrating a perceptual information infrastructure with robotic avatars: A framework for tele-existence. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-99)*, October 1999.

[7] E. Menegatti, F. Nori, and E. Pagello. Design of omnidirectional mirrors for mobile robotics based on robot's task. In *Submitted to an International Journal*, 2002.

[8] E. Menegatti, F. Nori, E. Pagello, C. Pellizzari, and D. Spagnoli. Designing an omnidirectional vision system for a goalkeeper robot. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup-2001: Robot Soccer World Cup V.*, pages pp. 78–87. Springer, 2002.

[9] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[10] A. Nakazawa, H. Kato, S. Hiura, and S. Inokuchi. Tracking multiple people using distributed vision systems. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA2002)*, pages pp. 2974–2981, May 2002.

[11] B. D. Olsen. Calibrating a camera network using a domino grid. *Pattern Recognition*, 34(5), May 2001.

[12] E. Pagello, M. Bert, M. Barbon, E. Menegatti, C. Moroni, C. Pellizzari, D. Spagnoli, and S. Zaffalon. Artisti veneti 2002: evolving an heterogeneous robot team for the middle-size league. In G. A. Kaminka, P. U. Lima, and R. Rojas, editors, *RoboCup-2002: Robot Soccer World Cup VI.*, L. N. on A. I. Springer, 2002.

[13] H. Takeda, N. Kobayashi, Y. Matsubara, and T. Nishida. A knowledge-level approach for building human-machine cooperative environment. In A. Drogoul, M. Tambe, and T. Fukuda, editors, *Collective Robotics*, volume 1456 of *Lecture Notes in Artificial Intelligence*, pages pages 147–161. Springer, 1998.

[14] G. Weiss. *Multiagent Systems*. MIT Press, 1999.