A Telepresence and Teleoperation Implementation on the Nao Humanoid Robotic Platform

Vineet Nagrath
Laboratoire Le2i, UMR
CNRS 5158, IUT, Le Creusot, France
Vineet.nagrath@gmail.com

Lukas Fuler
Vibot Master Student
University De Bourgogne
Le Creusot, France
Lukas.fuerler@gmail.com

Anmir Saeed Malik
University Technology Petronas,Perak, Maleysia
Aamir_saees@petronas.com.my

Fabrice Meriaudeau
Laboratoire Le2i, UMR
CNRS 5158, IUT, Le Creusot, France
Fabric.meriaudeau@u-bourgogne.fr

***Abstract*—** Telepresence and Teleoperation of robotic platforms expand the application domain of a robotic system. Unlike remotely operated robots, the telepresence systems built on autonomous platforms takes only a limited set of mission statements from the remote operator. The decisions and methods required for completing a mission on the workspace is still controlled autonomously by the robotic platform. We present such a system being implemented on the Nao humanoid robotic platform. Robot's search targets and end-actions are specified remotely by the user. Some components of the system are implemented on remote servers and are amalgamated using the cloud based service demand architecture. The system was modeled using Agent Relation Charts, which are components in the Computational independent layer of the Hyperactive Transaction Model, HTM5. *This is an extended version of the paper "An Auto Operated Telepresence System for the Nao Humanoid Robot" published in CSNT2013.*

Keywords- Na-Humanoid Robot; Agent Relation,Chart, Multiagent System, Telepresence, Teleoperation, Cloud Computing

I. Introduction

Telerobotics represents the area of robotics concerned with the control of robots from a distance, primarily employing wireless connections or the Internet for communication. It consists of two major subfields, namely Teleoperation i.e. to do work at a distance, and Telepresence i.e. to "feel" present at a remote location. Telerobotic systems play an important role in space exploration (e.g. NASA's Mars exploration rover), deep sea missions (e.g. repair of offshore oil platforms by marine remotely operated vehicles), robotic surgery (e.g. minimally invasive surgery employing microscopic manipulators) or in general in environments that are too hazardous to be directly accessed by humans (e.g. for handling radioactive materials or disarming bombs).

DOI-10.18486/ijcsnt.2012.1.1.01
ISSN-2053-6283

The aim of the work described here is to build an auto-operated, multi-agent telepresence system for the Nao humanoid robot using a structured model-driven design approach. This document will present the development

process of the system in the following structure: The remainder of the introductory section provides an overview of the Nao humanoid robot, the central actor in the system. The next section defines the project objectives, followed by a section that presents related work in the field of Telerobotics and outlines the ARC design methodology applied in the project. Section IV is the central part of this document explaining the development process of the telepresence system from initial concept over design to final implementation. Section V presents the project outcome by showcasing the implemented system, and finally section VI mentions some concluding remarks and provides pointers to possible future work.

A. Nao Humanoid Robot

Nao, depicted in "Fig. 1", is an autonomous humanoid robot designed and manufactured by the French startup company Aldebaran Robotics [1]. Its development began in the year 2004 and the first academic edition for research and education purposes was released in 2010. Nao is 58 centimeters tall and weighs around 4.3 kg. It has a total of 25 degrees of freedom and is equipped with a wide range of sensors including 2 HD cameras, 4 microphones, a sonar range finder, 2 IR emitters and receivers, 1 inertial board, 9 tactile sensors and 8 pressure sensors. Nao is programmed via the NAOqi framework, a cross-platform and cross-language software development kit, running on top of OpenNAO, the GNU linux-based robot operating system. The robot's functionalities can be controlled remotely over WiFi or Ethernet by means of proxy modules which automatically handle the network communication for the application programmer.

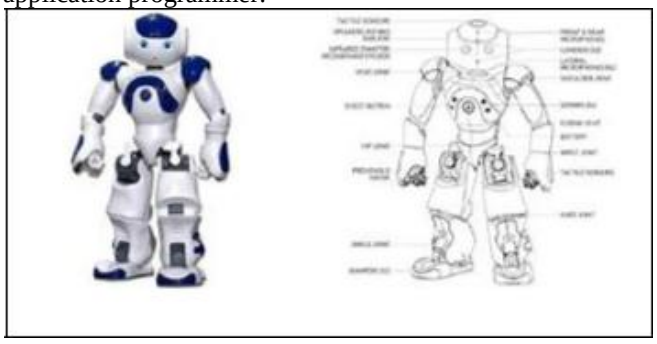


Figure 1. Frontal View of Nao robot. Image credit to [1].

II. OBJECTIVES

The primary project objective is to build a telepresence system that enables a remote operator to execute tasks via a Nao robot in a controlled environment. A task consists of reaching an operator-defined target location within the workspace, executing a suitable action depending on whether the location could be reached or not, and reporting back the requested feedback (such as a picture of the remote location) to the operator. The system should navigate the robot autonomously through the workspace avoiding static obstacles on the way, and it should be accessible from any Internet-connected client device world-wide. Since the central element of the telepresence system is the autonomous navigation module, a preliminary experiment has been conducted to assess Nao's walking accuracy with the conclusion that the robot's motion is not precise enough to rely on it[1]. This gives rise to the second objective, namely to design a closed-loop control system for correcting and controlling Nao's ground locomotion, which is to be integrated into the telepresence system.

The third objective is to develop the telepresence system following the outlines of the Agent Relation Charts (ARCs), the top layer of a new model-driven design methodology for cloud-connected multi-agent systems conceived at the CNRS Laboratory Le2i in Le Creusot (France) [2]. The project acts as a case study for the methodology aimed at acquiring subjective feedback about the applicability and usefulness of the model in the context of a real system implementation. The concepts of ARC modeling are discussed in the next section.

III. Related Work

This section showcases related research efforts in the field of telerobotics and introduces the ARC design methodology employed to develop the project.

A. Research in Telerobotics

The Telerobotics/Telepresence field is a vast research area with a multitude of open problems still to be solved. Consequently the research efforts vary from finding theoretical solutions over devising new architectures up to implementing real world systems aimed at serving a particular purpose. For example Bao et. al developed a multi-agent robot Tele-supervision architecture applied to the task of hazardous materials detection [3]. Their system was based on a three-level control hierarchy that allows instructing a robot remotely to execute a certain mission autonomously with the possibility of manual override in cases of emergency. Another example is the work of Budihal et. al , who developed a next generation experiential telepresence system with the aim of making the remote user experience

---

When instructed to walk 1m straight forward Nao missed its expected location on average by 30cm. According to the Aldebaran tech support this is the best accuracy the robot can currently achieve more immersive when compared to conventional 2-way audio-video systems [4]. Their system consists of a PRATHAM humanoid robot acting as the remote user's proxy for social interaction, a cognitive intelligence platform as a central knowledge gathering point, and an experience center that offers an advanced augmented reality control interface to the operator. Rekleitis et. al on the other hand developed a long-distance remote observation system suited for exploration and assessment of underwater environments by means of a Ramius amphibious robot [5]. As a

DOI-10.18486/ijcsnt.2012.1.1.01
ISSN-2053-6283

demonstration of the system the authors conducted a Tele-learning experiment in which the robot deployed at seaside in Barbados was accessed from a classroom in Canada. Their work served as a proof of concept of robot tele-supervision through web-based services and tele-learning as a means to awaken students' interest in the field of robotics.

A substantially different problem was tackled by Ding et. al, who studied an optimal approach to path planning and coordination of multiple remote operated Unmanned Aerial Vehicles (UAVs) for the task of protecting a convoy of ground vehicles [6]. The authors provide concise mathematical models for the path planning strategies applied to the protection of stationary and linearly moving ground vehicles, however freely moving vehicles are not yet supported in their model. As a last example, Santos et. al proposed a multi-agent system for robot Teleoperation based on a shared notion of reality between all participating entities
[7]. The main contribution of their work was to merge the agents' heterogeneous reality representations into a common format such that the collective knowledge can be shared efficiently. Several aspects of the systems found in the literature, such as optimal path planning, autonomous navigation and an Internet-based client interface were brought together to create the telepresence system for Nao.

B. *Agent Relation Charts*

Agent Relation Charts are modeling and specification tools for software development in cloud connected multi agent systems with a focus on the way human transactions take place [2]. ARCs are designed to be simple such that all stakeholders of a project can actively participate in its planning process. The primary actor in the ARC methodology is an agent, which is any entity that is able to carry out a transaction i.e. to adopt certain functionality within the system. Agents can be passive (static predefined behavior), active (dynamic adaptation to environment) or hyperactive (exert control on other agents). The most abstract top layer of the ARC Meta model relies on the five design views shown in "Fig. 2": The Relational and Structural views describe the individual agents in the system and the relationships among them. The Trade view shows the services/demands of the agents and the Hyperactivity view is used to model the control hierarchy in the system (hyperactive agents control others). Finally the Behavioral view is comprised by Use Case and Sequence ARC diagrams aimed at modeling how the system behaves in different usage scenarios.

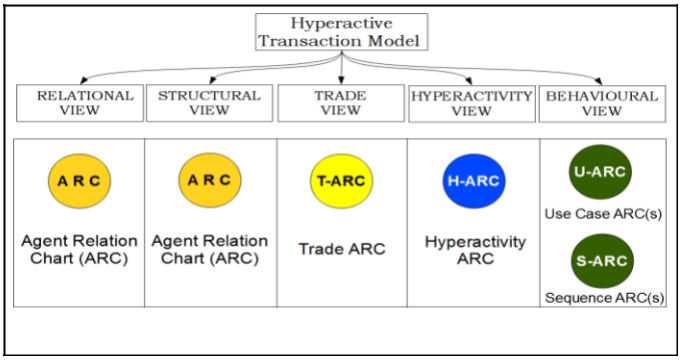


Figure 2. Five Design Views Of The ARC Meta Model. Image Credit To [2].

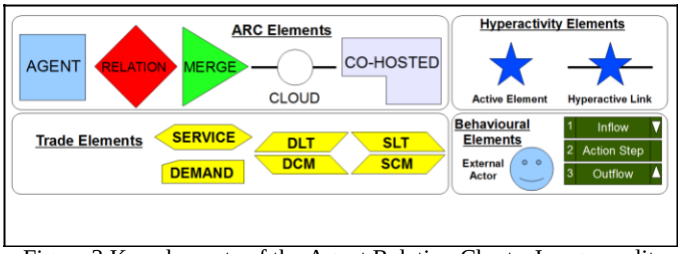


Figure 3.Key elements of the Agent Relation Charts. Image credit to [2].

The nomenclature and graphical rendering of the ARC design elements have been defined rigorously with the goal of ensuring consistency among the different diagram types. The elements used in the top layer are depicted in "Fig. 3". As can be seen the symbols are divided into four groups according to diagram type, however just the key elements situated in the top left corner are described here. Agents (blue squares) are entities that execute/access different actions according to a built-in behavioral logic. Relations (red rhombuses) are the connection points between agents or other relations; they store and manage the variables that govern the mutual behavior of the connected entities. Merges (green equilateral triangles) are special relations whose task it is to manage the information received through their input links and provide it in a combined/processed format to other agents. Clouds (white circles) are uniquely identifiable networks that establish the communication between entities. Co-hosted Regions (purple polygons) are used to group entities that are hosted on the same physical device.

IV. System Development

This section describes the development stages of the telepresence system for Nao. Firstly, the system concept is introduced. Then the ARC high level design, the UML medium level design and the algorithms employed in the implementation are explained.

A. *System Concept*

For the sake of simplifying the navigation procedure the robot workspace is assumed to be a flat, rectangular segment of floor located in a controlled indoor environment large enough to allow for free movements. As illustrated in "Fig. 4" such an environment is most efficiently observed using an array of downward-facing top-view cameras

DOI-10.18486/ijcsnt.2012.1.1.01
ISSN-2053-6283

mounted above the workspace. The robot location within this 2D map (single plane in 3D)

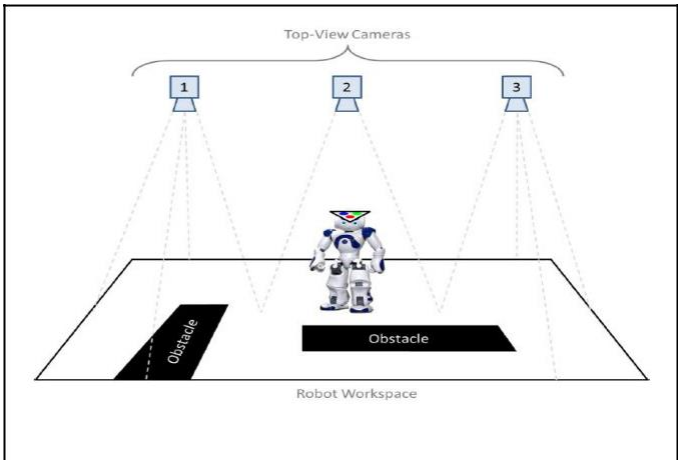


Figure 4. Schematic Overview Of The Robot Workspace Showing Nao, Top-View Cameras Used For Localization And Static Obstacles On The Floor.

can be represented by three parameters: The X and Y coordinates and the heading angle.

In order to distribute the responsibilities within the sys-tem over several physical hosts the following five software agents are defined: A Cam Reader agent processes the images acquired by a single camera and provides the relative robot location to the Manager PC. The Manager PC is the central control unit that executes tasks by localizing the robot globally and instructing it where to go and what to do. The Nao agent represents the control and communication software running on the robot, responsible for providing an interface for accessing the robot's functionalities. The Web Server agent is the communication point between Manager PC and remote clients, and hence also provides the control interface to the remote operator. Lastly, the Remote Client agent is an abstract representation of any Internet-capable client device that connects to the Web Server agent.

The "Fig. 5" provides an overview of the telepresence system. The Cam Reader, Nao and Manager PC agents are collaborating locally at the site of the system setup to execute a robot task. They are interconnected via a Local Area Network (LAN). The Web Server and Remote Client agents can be located anywhere and are connected to the local agents via Internet.

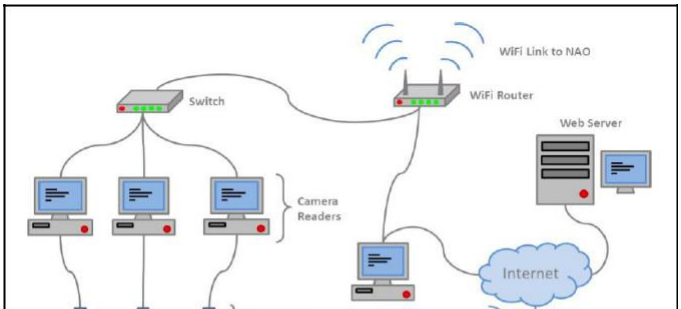


Figure 5. Overview Of The Telepresence System Depicting The Individual Agents, Hardware And Communication Lin

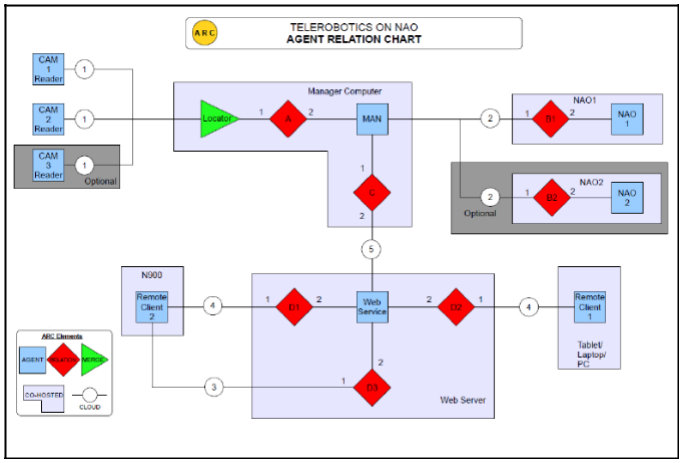


Figure 6.ARC diagram representing Structural and Relational views

B. *ARC-High Level Design*

All ARC diagrams required to model the different aspects of the telepresence system were designed according to the specifications of the hyperactive transaction model, however due to the limited extent of this paper only two of the primary diagrams are given here. As can be seen in "Fig. 6" the Structure-Relation diagram has a similar structure to the system topology shown in "Fig. 5". The Cam Reader agents on the top left are connected to the Locator merge block responsible for merging the relative locations into a global one. The Manager PC is connected to two Nao agents; however the second one is marked as optional since the system currently only supports a single robot. The Web Server has several connections to remote clients, which suggests that it is serving multiple clients simultaneously. The individual cloud connections between the agents are

numbered in order to identify them; same number means same network. The co-hosted region in the center of the diagram encloses the Manager agent and the Locator merge, which collectively represent the Manager PC. Relations with the same character identifier but a different number (e.g. B1 and B2) are instances of the same relation that are just customized for different instances of the same agent (e.g. Nao1 and Nao2).

The Trade-ARC shown in "Fig. 7" was used to model the trade logic that governs the system's service-demand behavior. It describes the services/demands that the individual agents offer/require. As can be seen in the diagram the Remote Clients offer target location and action

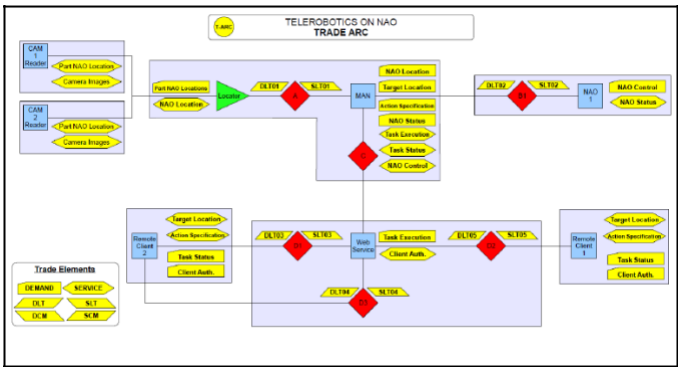


Figure 7. ARC Diagram Representing The Trade View

specification given by the human operator to the MAN agent which needs this information to execute the task. They require an authentication service offered by the Web Server and the status of task execution offered by the MAN agent. The Web Server, once received a new task from the remote client, requires the task execution service offered by the MAN agent. The MAN agent needs the current location of Nao and its status in order to control it. The global Nao location is offered by the Locator, which demands the partial locations from the Cam Readers. Note that the DLT, SLT, DCM and SCM elements are not actively used within this project, since all the service-demand associations in the system are fixed and do not depend on the trade economy.

C. *Implementation*

The system implementation was entirely carried out in Python [10], a high-level cross-platform interpreted programming language offering a wealth of built-in libraries and a rich syntax. All image-related processing was handled by the Open Source Computer Vision library (OpenCV [11]), and the communication with the Nao robot was done by interfacing its functionalities through the NaoQi framework. The Web Server agent was deployed online on Google App Engine [12], a cloud computing framework for developing and hosting web applications in Google-managed data centers. For the Cam Reader and Manager PC agents a simple command line interface was implemented, since these two applications are solely used by the system administrator. However in order to provide some monitoring facilities to the administrator, the current camera frames and the path followed by the robot are displayed in OpenCV windows in the course of task execution. The Web Server on the other hand provides an interactive HTML interface to the remote clients which can be accessed worldwide through any standard web browser at HTTP://NAOTELEROBOTICS.APPSPOT.COM/.
The main algorithms developed in the course of the implementation process are shortly outlined in the following sections.

D. *Algorithms and Methods*

In order to localize Nao within the view of a single camera the color-based marker detection technique summarized in "Fig. 8" was applied. A triangular marker containing a circle in one of the three primary colors (red, green and blue) in each corner has been mounted on top of Nao's head such that it can be clearly seen by the top-view cameras. The largest blob in each of the masks is selected and used to compute the geometrical information of the marker: The marker center corresponds to the center of gravity of the three color blobs and the orientation can be obtained by calculating the slope of the line connecting the marker center and the center of the red blob (assumed to be the front).

Using the method described above the 2D pixel location of the marker and its orientation can be obtained. However, as illustrated in "Fig. 9", this location does not correspond to the real robot location on the ground. The problem arises because the plane at which the marker is detected is situated at a different depth from the camera w.r.t. the plane on which

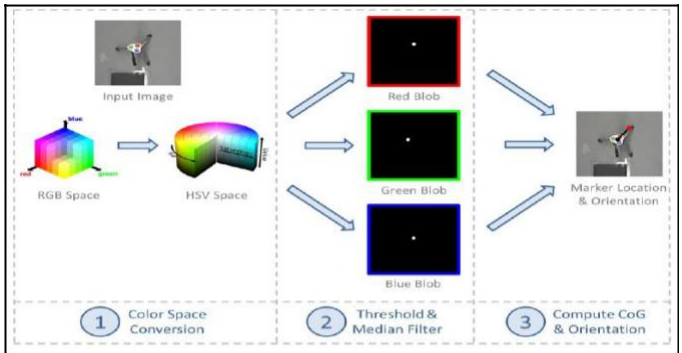


Figure 8. Marker Detection Procedure Used For Localizing Nao Within A Single Camera View

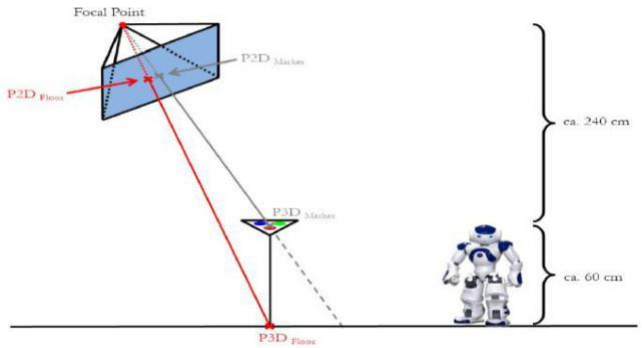


Figure 9. Schematic Workspace View Depicting The Difference Between The Detected Marker Position And The Real Robot Location

DOI-10.18486/ijcsnt.2012.1.1.01
ISSN-2053-6283

Nao is standing. Since accurate robot localization is the key factor for the success of the telepresence system a correction procedure was implemented as explained next.

In the telepresence system the cameras are mounted on the ceiling facing downwards, i.e. the Z-axis extending from the cameras is vertical. Since the cameras' positions are fixed and Nao has a constant height when standing and walking, the vertical distance between camera marker and camera floor can be measured. Given these two distances and the camera projection matrix (obtained by calibrating the camera using OpenCV's built-in chessboard pattern calibration facilities) the method tries to estimate the robot's pixel location on the floor ($P2D_{Floor}$) starting from the pixel location of the marker ($P2D_{Marker}$) in the following way: $P2D_{Marker}$ is multiplied with the inverse of the camera matrix yielding a 3D point on the ray extending from the focal point through $P2D_{Marker}$. This point is then normalized and multiplied with the marker distance from the camera, which gives the 3D location of the marker's center point ($P3D_{Marker}$). Then the Z coordinate of the point is set to the floor distance from the camera, i.e. the point is "moved down" to the floor. The obtained $P3D_{Floor}$ is then projected back onto the image plane by pre-multiplying it with the camera matrix which yields $P2D_{Floor}$, the real robot location within the image.

In order to obtain a global map of the workspace the individual camera views were combined as illustrated in "Fig. 10". The global map was created by rotating the camera views by 90 and then merging them manually. The offsets of the camera views w.r.t. the global map were measured and included in the XML configuration file of the Manager PC agent, which applied a simple transformation to the relative camera-based coordinates to obtain the coordinates within the global map.

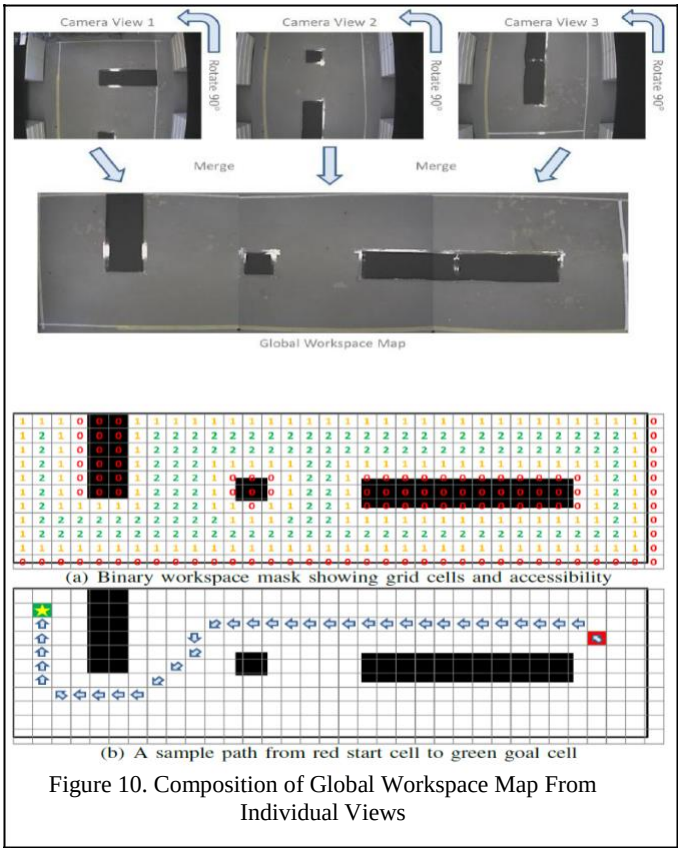


Figure 10. Composition of Global Workspace Map From Individual Views

In order to direct the robot from any valid start location within the workspace to an operator-defined target location an optimal path connecting the two points has to be planned. Since doing path planning at pixel level is not feasible (due to the robot's walking inaccuracies and size) the workspace map has been subdivided into the cell array depicted in "Fig. 11". Each cell is assigned a value (0 for obstacle, 1 for dangerous because close to obstacle, 2 for accessible) depending on its location within the map. In order to find an optimal path within the described cell matrix, a customized version of A*, an efficient graph traversal algorithm invented in the year 1968 at the Stanford Research Institute, is employed. The cost of a node within the path is computed as the weighted sum of four factors: The cost to reach the previous node along the path, the distance of the move to reach the current node, a factor penalizing the change of direction (since Nao takes time to do so), and a factor penalizing the access of dangerous cells (in order to keep Nao at a safe distance from obstacles). A sample path computed by the A* planner is shown in "Fig. 11(b)".

E. UML based medium level design

Since at the time of developing this project just the top layer of the ARC methodology was finalized, UML class diagrams "Fig. 12" were chosen as the platform-independent medium layer of the system design. The planning of the Naoagent was not necessary, since essentially it represents the public interface to access its functionalities, which is readily available through the NaoQi framework running on the robot. Also, the design of the Remote Client was not required, since it is an abstract representation of the operator's web-browser, an essential piece of software available on any Internet-capable device. Consequently only the UML modeling of the Manager PC, Cam Reader and Web Server agents was carried out.

The main responsibility of the Cam Reader agent is to obtain the relative location of Nao within the frames fetched from the associated top-view camera and provide that information to the Manager PC. The class diagram depicted in "Fig. 12" shows how this behavior was implemented in terms of four regions responsible for different aspects of the functionality. There are classes for communicating with the camera, for serving the current camera image and relative robot location to the Manager PC, for processing the images, and for managing the agent configuration in terms of XML files. The central class is the CamReader, which combines the functionalities of all other classes and is the one launched by the system administrator.

The Manager PC is the central control unit, responsible for coordinating Nao's movements and directing it to the target location; its UML class architecture is shown in "Fig. 13". As before the agent's behavior has been split into four regions. There are classes for global robot localization and path planning, for managing a FIFO (First In First Out) queue of operator-issued tasks, for communicating with

Web Server, Nao and Cam Reader agents, and for maintaining an XML configuration that contains all the information necessary to instantiate a Manager PC agent. The primary advantage of using configuration files is that the agent's behavior can be adapted to a new system setup by simply modifying the values in the XML file and no changes to the code are required.

The architecture of the Web Server agent depicted in "Fig. 14" is substantially different from the two previous ones, since it was designed to be hosted on a cloud-based infrastructure online which does not allow full control over the system's resources. In particular the applied design principle of CGI-based applications is Inversion of Control (IoC), i.e. the application developer only needs to provide the handler code for each functionality and define the *hURL*, *Handler.mappings*.

All the HTTP related issues are handled by the server environment and the application programmer is offered a flexible API to process incoming client requests and send back dynamic contents or HTML pages. The UML diagram consists of three main elements, namely the abstract *ClientBrowser* on the left, the *ManagerPC* agent on the right and the *WebServer* delimited by the gray box in the center. Since there is no direct communication between client and Manager PC, the Web Server acts as mediator between the two parties responsible for forwarding client requests to the manager, and providing feedback on task execution to the client. The red part in the diagram is responsible for generating HTML pages containing the user interface for the clients and for providing updates on task execution. The green part is communicating with the Manager PC, mainly for fetching new task definitions and updating task-related information. The yellow part in the center is a database persistence layer used to store the robot's path, update messages, robot camera images and task definitions in between requests.

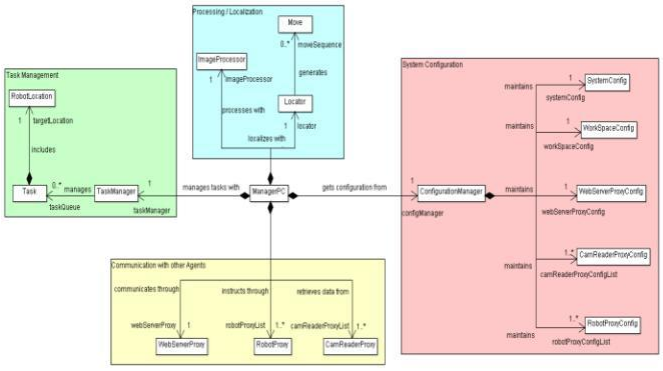


Figure 12. UML Class Diagram For *Managerpc* Agent

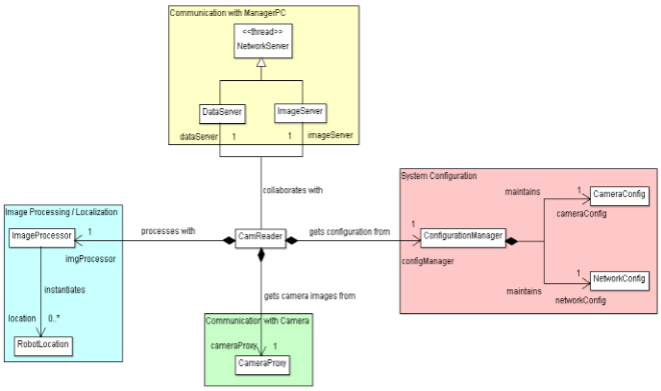


Figure 13. UML Class Diagram For *Camreader* Agent

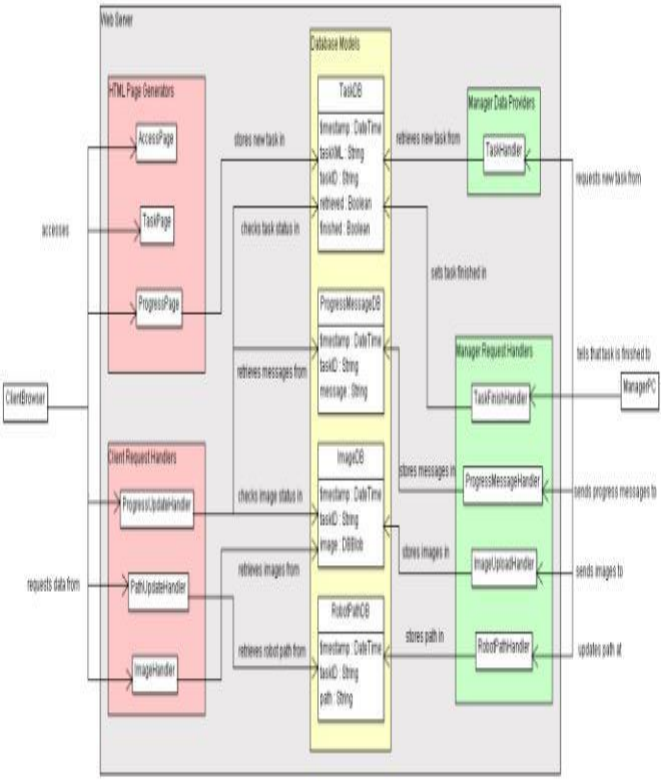
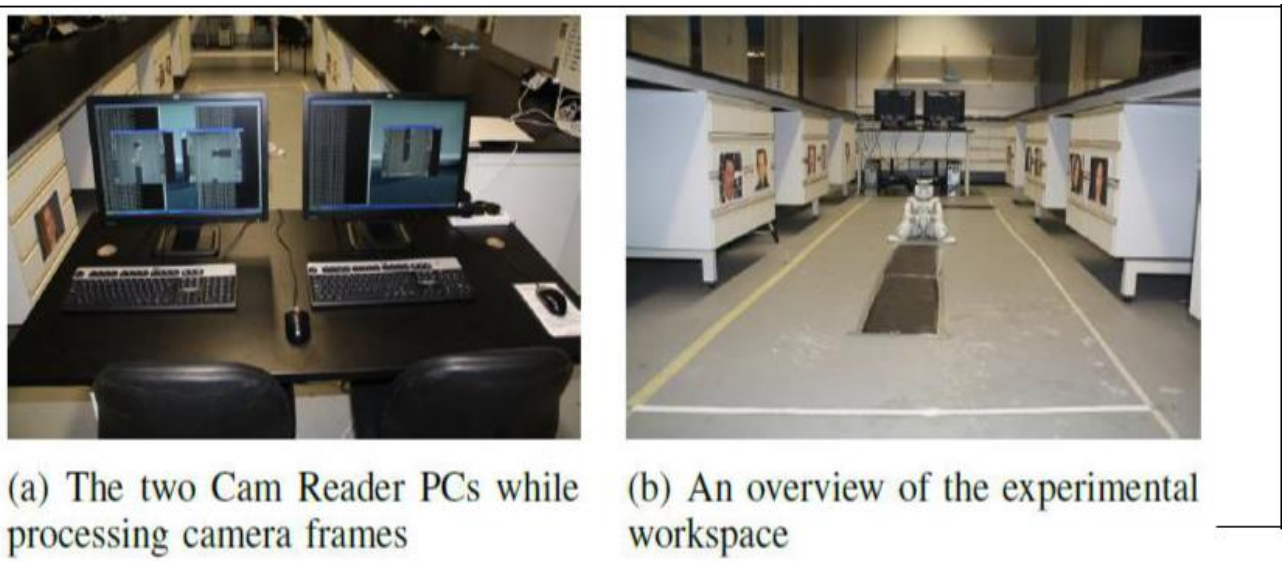


Figure 14. UML Class Diagram For *Webserver* Agent



(a) The two Cam Reader PCs while processing camera frames

(b) An overview of the experimental workspace

Figure 15. Composition Of Global Workspace Map From Individual Views

V. Experimental Results

The experimental setup at the UTP laboratory shown in "Fig. 15" was exactly built according to the schematic overview depicted in "Fig. 5". In order to assess the performance of the telepresence system an array of experiments has been conducted. In most of these experiments Nao was able to successfully reach the target location and complete the given task. However in some cases the execution could not be terminated, because Nao tended to overshoot the specified target location and hence had to turn around (a full 180 rotation) in order to come back. Since Nao is unable to turn on the spot, this rotation caused an unintentional change in location and hence it missed the location again. The result of this was that the robot was oscillating closely around the final location but was unable reach it exactly. The problem was resolved by specifying a tolerated distance of half a cell (=7.5 cm) to the target location.

After completing the Web Server agent the next testing stage was started by conducting the following cross-continent experiment: An operator issued three different tasks through the remote client interface from his laptop located at the Le2i laboratory in Le Creusot, FRANCE and all of them were completed successfully by the robot in the UTP laboratory, MALAYSIA. This showcases the location and device independence of the telepresence system, i.e. as long as the remote operator has an Internet connection he/she can control the system from anywhere in the world.

VI. Conclusions & Future Work

A multi-agent telepresence system able to autonomously navigate a Nao humanoid robot to an operator-defined target location within a static environment has been successfully implemented. A closed-loop gait control paradigm for the Nao robotic platform has been devised and proven to be effective, i.e. despite the significant inaccuracy in the robot's motion it was able navigate to the target location while safely avoiding obstacles. The workspace was observed by three top-view cameras which were used to localize the robot using a color-based marker detection technique. An adapted version of the A* search algorithm was applied to plan an optimal path from current to target position. The conducted experiments show that the system is operative and that it satisfies all usage scenarios planned during the ARC design stage. The system was developed using a structured multi-stage approach, employing ARC diagrams for abstract high level planning, UML class diagrams for agent modeling and Python in combination with several libraries as implementation language. The ARC design methodology has proven to be a valuable tool to express the conceptual system design and to model its most relevant aspects in a consistent platform-independent manner. However just the top layer was formulated at the time of creating this project and hence a comprehensive evaluation of its applicability and efficiency can only be done once all model layers have been finalized. The experiences with the ARC methodology made throughout the project were reported to the authors [2].

One possible direction for future work could be to enable the collaboration of multiple robots within the workspace, which would allow for the formulation of more sophisticated tasks. Such an extension would involve the design of a coordination strategy for the robots and the development of a new marker detection technique, since several distinct markers would need to be detected simultaneously. Another improvement of the system could be to make the telepresence experience of the remote user more immersive by e.g. including a live view of the robot's camera in the web-based interface. It would also be useful to develop tools that aid the system administrator in doing the initial setup by carrying out some tasks such as the intrinsic camera calibration or the global map composition automatically.

DOI-10.18486/ijcsnt.2012.1.1.01
ISSN-2053-6283

NOTE: This is an extended version of the paper "An Auto Operated Telepresence System for the Nao Humanoid Robot", made available on the request of the original Publisher. The Initial paper was published in CSNT2013.

References

[1] A. Robotics, "Aldebaran Robotics - Company," Paris, France, 2012.[Online]. vailable:http://www.aldebaran-robotics.com/ 1

[2] V. Nagrath, F. Meriaudeau, A. Saeed Malik, and O. Morel, "Agent Relation Charts (ARCs) for Modeling Cloud based transactions," in Proceedings of the International Conference on Communication Sys-tems and Network Technologies. Rajkot, India: Laboratoire Le2i, UMR CNRS 5158, IUT, Le Creusot, FRANCE; University Technology Petronas, Perak, MALAYSIA, 2012, p. 6. 2, 3, 4, 7

[3] J. Bao, Y. Guo, A. Song, and H. Tang, "A Multi-Agent Based Robot Telesupervision Architecture for Hazardous Materials Detection," in Proceedings of the 2010 IEEE International Conference on Information and Automation, Harbin, China, 2010, pp. 2428–2432. 2

[4] R. Budihal, N. Mohanan, S. A. Anand, and S. S. Kamat, "Exploration and Implementation of a Next Generation Telepresence System,"Ban-galore, India, pp. 1–6. 2

[5] I. Rekleitis, G. Dudek, Y. Schoueri, P. Giguere, and J. Sattar, "Telep-resence across the Ocean," in Canadian Conference on Computer and Robot Vision, Montreal, QC, Canada, 2010, pp. 1–8. 2

[6] X. C. Ding, A. R. Rahmani, and M. Egerstedt, "Multi-UAV Convoy Protection : An Optimal Approach to Path Planning and

[7] V. Santos, P. Santana, L. Correia, and J. Barata, "Teleoperation mecha-nisms in a Multi-Agent System," Lisbon, Portugal, pp. 170–176, 2008. 2

[8] L. Furler,¨ A. Saeed Malik, F. Meriaudeau, and V. Nagrath, "An Auto-Operated Telepresence System for the Nao Humanoid Robot,"MSc Thesis, Universiti Teknologi Petronas, 2012. [Online]. Available: http://naotelerobotics.shorturl.com 3

[9] "OMG Unified Modeling Language (OMG UML), Infrastructure,"Availablehttp://www.omg.org/spec/UML/2.4.1/ Infrastructure 4

[10] G. Van Rossum, "Python Programming Language," Amsterdam,1989. [Online].

[11] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools,2000.[Online].Available: http://opencv.willowgarage.com/wiki/ Welcome 5

[12] D. Sanderson, Programming Google App Engine: Build and Run Scal-able Web Apps on Google's Infrastructure, 1st ed., M. Loukides,Ed. Sebastopol, CA: O'Reilly Media, Inc., 2010.

DOI-10.18486/ijcsnt.2012.1.1.01
ISSN-2053-6283