ARE: Augmented Reality Environment for Mobile Robots

Mario Gianni(✉), Federico Ferri, and Fiora Pirri

ALCOR Laboratory, DIIAG, Sapienza University of Rome, Rome, Italy
gianni@dis.uniroma1.it

Abstract. In this paper we present ARE, an Augmented Reality Environment, with the main purpose of providing cognitive robotics modelers with a development tool for constructing, at real-time, complex planning scenarios for robots, eliminating the need to model the dynamics of both the robot and the real environment as it would be required by whole simulation environments. The framework also builds a world model representation that serves as ground truth for training and validating algorithms for vision, motion planning and control. We demonstrate the application of the AR-based framework for evaluating the capability of the robot to plan safe paths to goal locations in real outdoor scenarios, while the planning scene dynamically changes, being augmented by virtual objects.

1 Introduction

Augmented Reality (AR) has recently stepped beyond the usual scope of applications like machine maintenance, military training and production. AR is a compelling technology allowing cognitive robotics developers to design a variety of complex scenarios involving real and virtual components. Indeed, programming complex robotic systems, operating in dynamic and unpredictable environments, is a challenging task, when it is required to go beyond laboratory experiments. Furthermore, as the complexity of the robotic system is inflated by an increase of robot components and functionalities, testing the whole set of interconnections between hardware and software components becomes exponentially difficult. The cognitive roboticists are more and more exposed to hard problems requiring a great amount of possible real world situations that are difficult to predict, when most of the test require laboratory experiments. In this context, Augmented Reality (AR) facilitates robot programming, providing to robot developers a technology for testing the robot system which is much more flexible than a complete simulation environment, as it allows developers to design a variety of scenarios by introducing any kind of virtual objects into real world experiments. AR-environments can facilitate experiments bypassing complex simulation models, expensive hardware setup and a highly controlled environment, in the various stages of a cognitive robot development.

In this paper, we present an AR-based simulation framework which allows robot developers to build on-line an Augmented Reality Environment (ARE)

A. Natraj et al. (Eds.): TAROS 2013, LNAI 8069, pp. 470–483, 2014.
DOI: 10.1007/978-3-662-43645-5_48,

for real robots, integrated into the visualization interface of Robot Operating System (ROS) [1]. The system we propose goes beyond an interface for drawing objects, as the design exploits a stochastic model activating the behaviors of the introduced objects. Objects, people, obstacles, and any kind of structures in the environment can be endowed with a behavior; furthermore, a degree of certainty of their existence and behaviors, with respect to what the robot perceives and knows about its space, can be tuned according to the experiment needs.

To illustrate the advantages of an AR based simulation framework for robot design and experiments, as opposed to a complete simulation framework in scenario design and test, and to show the benefits of our approach, we set up several path-planning experiments, with increasing complex environments. In particular we show that the framework allows to compare the performance of the path-planner with respect to several parameter sets in any simple outdoor settings augmented with ARE.

The paper is organized as follows. In the next Sect. 2 we discuss related work highlighting the novelty of our approach. In Sect. 3 we describe the main components of the AR-based simulation framework, detailing on the dynamic model. Section 4 describes the robot setup and reports the results obtained by a paradigmatic application of ARE for evaluating the capabilities of the robot in navigation tasks.

2 Augmented Robot Reality: State of the Art

Augmented Reality (AR) is a recent emerging technology stemming from Virtual Reality (VR). AR develops environments where computer-generated 3D objects are blended (registered) onto a real world scene. The AR computer interfaces have three key attributes: (1) they combine real and virtual objects, (2) the virtual objects appear registered in the real world and (3) the virtual objects can be interacted in real time [2]. The main difference between AR and VR is that AR eliminates the need to model the entire environment as it supplements the real world instead of replacing it. AR research has been active in robot applications such as maintenance [3], manual assembly [4] and computer-assisted surgery [5]. Several works have also addressed robot programming. One of the first robotic applications of AR was Milgram *et al.* application in telerobotic control [6]. In telerobotic applications an operator is provided with visual feedback of a virtual wireframe robot superimposed over an actual physical robot located at its remote working environment [7]. The virtual robot would execute a task for evaluation by the operator and the task would be transferred to the real robot if it is satisfactory. In robots monitoring [8,9] the visualization of complex data is overlayed onto the real world view. Namely, a view of the robot and the environment is synthesized graphically from on-board sensory data, such as camera images, and presented to remote operators, increasing their situation awareness. In [10] the authors present an AR-based approach for intuitive and efficient programming of industrial robots. Here, tool trajectories and target coordinates can be visualized and manipulated by means of an interactive laser projection. Bischoff

and Kazi [11] report an AR-based human robot interface for applications with industrial robots. Their work focuses on visualizing workflows that could help inexperienced users to cope with complex robot operations and programming tasks. AR has also been used to program painting robots [12]; here a spray gun, which is hand-held by a user, is tracked and the corresponding virtual spray is displayed to the user through a Head-Mounted Display (HMD). A feedback is obtained by tracking the object to be sprayed using markers, while the user movements are converted into a robot program. Daily *et al.* use robots AR to convey the robot state information: virtual arrows are overlaid on the robot to show its heading. Also bubblegrams displaying robot states and communications [13] have been used to improve the interaction between co-located humans and robots. Dragone *et al.* [14] too place virtual characters on top of real robots to express robot states through natural interaction modalities, such as emotion and gestures. Giesler *et al.* [15] apply AR for prototyping warehouse transport robotic systems. This application enables the user to interactively construct a topological map of paths and nodes in the warehouse by walking around and pointing at the floor. The robot is instructed to move along the map and the planned path is visualized to the user as an augmentation of the real world view. The user can inspect possible intersecting points with workers paths, thus obtaining an estimation of the efficiency of the solution. Stilman *et al.* [16] create an AR environment for decoupled testing of robot subsystems. Results from motion planning and vision algorithms are visualized over real world images provided by both external cameras and a camera mounted on the robot. Collett and MacDonald [17] apply AR for debugging robot applications. They overlay virtual information such as robot sensory and internal state data onto real world images of the robot environment, in so providing robot developers with a better understanding of the robot world view. Chong *et al.*, in [18], introduce an AR-based robot programming (RPAR-II) system to assist users in robot programming for pick and place operations with an human robot interface, enriching the interactions between the operators and the robot. In their system, a virtual robot, which is a replicate of a real robot, performs and simulates the task planning process.

Green et al. in [19] proved AR to be a suitable platform for human-robot collaboration. Similarly, Billinghurst *et al.*, in [20], show that AR allows to share remote views (ego-centric view) visualizing the robot view, to the task space (exo-centric view). Furthermore, they show that AR can provide support for natural spatial dialog by displaying the visual cues necessary for a human and a robot to reach common ground and maintain situation awareness. Billinghurst and colleagues show also that the robot can visually communicate its internal states to its human collaborators, by graphic overlays on the real world-view of the human [21].

3 The AR-Based Simulation Framework

The AR-based simulation framework registers virtual objects, such as robots, cars, people, pallets and other kind of obstacles into the real environment

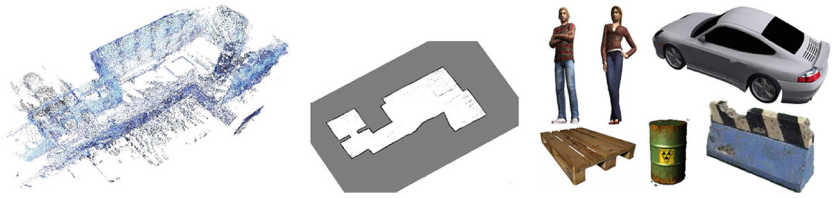


Fig. 1. On the left a 2D and 3D maps for the ARE environment representation, on the right 3D models of virtual objects.

(see Fig. 1). The model for the life-cycle of each virtual object is stochastic. It formally structures the interaction between the virtual objects and the real environment by both avoiding collisions and handling occlusions effects. The AR-based simulation framework includes: (1) the model of the real environment, (2) the model of the virtual objects, namely *artefacts* and, (3) the AR-builder server.

3.1 The Real Environment Representation

The real environment representation concerns both the 2D occupancy grid map $\mathcal{M}_{2D}$ and the octree-based 3D map $\mathcal{M}_{3D}$ on the left of the panel in Fig. 1, [22–24]. This representation has been proved to be compact and easy to update incrementally, thus accounting for dynamic obstacles and changing scenes. In fact, a polygonal mesh $\mathcal{S}_E$ is used to geometrically represent the environment. The polygonal mesh renders, together with the basic environment structure, also the 3D models of the artefacts, so as to correctly place the artefacts in the real scene, thus handling the occlusions with the existing objects [25].

3.2 The Artefacts Model

An artefact is a dynamic object defined by three main structures, illustrated in Fig. 2, bottom-right of the panel: the properties, selected by a mixture of Poisson distributions, a life cycle specifying the arrival and leaving time and, finally, a Markov decision process specifying the artifact main behaviors, given the life cycle and the selected properties.

We are given a space $\mathbb{Q}$ of all possible tuples defining an artefact type; actually we can think of $\mathbb{Q}$ as the set of all possible matrices $\mathbf{Q}$ where each row specifies a particular tuple Q of properties of an artefact A, with $Q = \{l, b, \mathbf{p}(.,t), \mathbf{q}(.,t), \Phi\}$. Here l is a label denoting the virtual object type, b is the bounding box of the artefact, and $\mathbf{p}(.,t)$ and $\mathbf{q}(.,t)$ denote the position and orientation operators of the artefact, taking temporal values in the stochastic Poisson model, described in the next sub-section. Φ is a set of additional features. Each artefact is geometrically specified by a polygonal mesh $\mathcal{S}_A$, representing the virtual object as a set of faces and a set of vertices, according to the face-vertex model, along with additional information such as color, normal vector and texture coordinates. We assume that the geometry of the mesh data structure is not morphable and

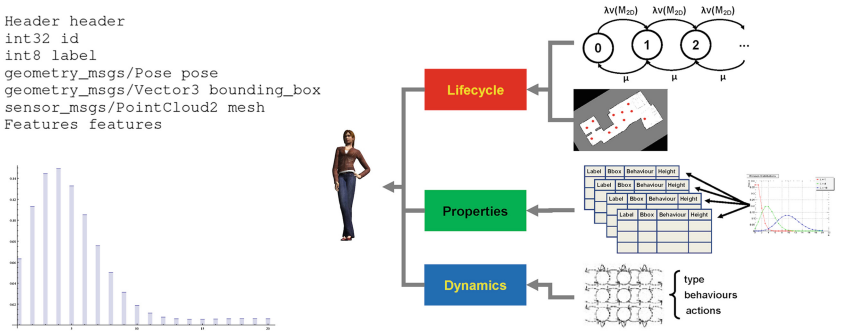


Fig. 2. On the left top the ROS message specifying an instance of an artefact. On the left, bottom, a Poisson Mixture of 5 components with parameters vector $(15, 7, 3, 21, 5)^\top$. On the right the artefact structure schema.

warpable so that the bounding box is fixed for its life-cycle. This representation allows the AR-builder server to manage the collisions between the artefacts populating the real environment. Within the framework, an artefact is implemented by a ROS message, see Fig. 2, top-left panel.

In this subsection we introduce the simulation model regulating the spatio-temporal behavior of the artefacts populating the real environment. Namely we introduce both how properties are sampled in $\mathbb{Q}$ and the arrival and leaving time of the artefacts. This model is based on a marked Poisson process [26–28], whose marks correspond to an identifier selecting a tuple from a matrix $\mathbf{Q} \in \mathbb{Q}$ and the artefact life-cycle. A marked Poisson process is a Poisson Process which has each point labeled with a mark. These marks are used to connect the artefact properties with its specific life-time cycle. Indeed, the model estimates the arrival and exit time of the artefacts as well as the labels governing the choice of the artefact properties.

Let the variable X, selecting a tuple Q in the space $\mathbb{Q}$, be a stochastic variable sampled from a mixture of n Poisson distributions, with n the number of object classes, namely:

$$Pr(X = k|\lambda_1, \ldots, \lambda_n) = \sum_{i=1}^{n} \pi_i \frac{\lambda_i^k}{k!} \exp(-\lambda_i) \tag{1}$$

Here $\sum_i \pi_i = 1$, λ_i, $i = 1, \ldots, n$ are the Poisson distribution parameters for each mixture component. Note that since n is given a priori from the classes of buildable objects (e.g. cars, pallets, girls, boys,..) parameters estimation is set by the EM [29]. The choice of the tuples is specified upon arrival of a group of artefacts. This is detailed below.

Let $M_{2D} \subseteq \mathbb{R}^2$ be the occupancy grid designating the flat structure of a generic environment taken as base of the representation (e.g. the courtyard of the department, a set of corridors). The arrival time t_a of artefact a, in M_{2D} can be assigned according to a Poisson process. Here arrival time $t \in [0, T]$ has

an average rate $\lambda \in \mathbb{R}$. The times between successive arrivals are independent exponential variables having mean $\frac{1}{\lambda}$. Upon arrival, an artefact is supposed to leave M_{2D} after a certain time. Similarly to arrival time, the time last of each artefact is an independent exponential variable with mean $\frac{1}{\mu}$, where $\mu \in \mathbb{R}$ is the average rate artefact leaving time. From these assumptions it follows that arrivals and leavings time t are ruled by a time-homogeneous irreducible, continuous-time Markov chain $\{N(t)|t \geq 0\}$ with state space $\mathbb{N}^+$. Let $\nu(M_{2D})$ be the area of M_{2D}, let $o(\Delta t)$ be an infinitesimal of a higher order than Δt[1], then the stationary transition probabilities $p_{ij}(\Delta t)$ between state i and state j, for infinitesimal lapses of time Δt, are given by

$$p_{ij}(\Delta t) := \mathbf{P}(N(t+\Delta t) = j | N(t) = i) = \\ = \begin{cases} \lambda\nu(M_{2D})\Delta t + o(\Delta t) & \text{if } j = i+1 \\ 1 - (\lambda\nu(M_{2D}) + \mu)\Delta t + o(\Delta t) & \text{if } j = i \\ \mu\Delta t + o(\Delta t) & \text{if } j = i-1 \\ o(\Delta t) & \text{if } |j-i| > 1 \end{cases}$$

$\{N(t)|t \geq 0\}$ can be seen as a special case of a birth-death process [30], with birth rates $\lambda_i = \lambda\nu(M_{2D})$ and death rates $\mu_i = \mu$, for each $i \in \mathbb{N}^+$. Let the arrivals of new artefacts in M_{2D} occur according to a Poisson process with intensity $\lambda\nu(M_{2D})$ and let, upon arrival, the lifetime of each artefact be independent and exponentially distributed with mean $\frac{1}{\mu}$, then $N(t)$ returns the number of artefacts alive at time t (see Fig. 3(a)). Given the life-cycle, artefacts populate the environment and, given the Poisson mixture, upon arrival they draw their specific properties, including features such as being a car or being a pallet, from the distribution. The final component of the artefact structure is its characteristic behavior determined by a motion planning algorithm that would govern what it can actually do in the scene. To model the specific chosen behavior a finite-horizon Markov Decision Process is introduced:

$$\mathcal{H}_A(t) = \{D, S, \mathcal{A}_s, p_t(\cdot|s,\alpha), r_t(\cdot|s,\alpha) : t \in D, s \in S, \alpha \in \mathcal{A}_s\}$$

Here $D = \{0, \ldots, d\}$ is the set of decision epochs, depending on the life time d of the artefact, $S \equiv M_{2D} \times \{0, \frac{\pi}{2}, \pi, \frac{3}{2}\pi, 2\pi\}$ is the set of poses that it can take on during its life cycle, $\mathcal{A}_s$ is the set of possible actions when the artefact is in the state s, $p_t(\cdot|s,\alpha)$ is the transition probability from state s at time t to the next state s' at time t' when action α is selected at the decision epoch t, and $r_t(\cdot|s,\alpha)$ is the reward earned when the artefact is in state s and action α is selected at decision epoch t. Actions are choices about both behaviors and motion planning. In other words an action α is a function $\alpha : \mathcal{B} \times \Pi \times S \mapsto (k_1, \ldots, k_m)^\top$ mapping a set of predefined behaviors $\mathcal{B}$, the set of motion planning algorithms Π, supporting the behaviors, and the set of states S, into a vector of numbers accounting for the choice. Behaviors specify what the artefact can do within the environment, i.e. walking, running, following another artefact, stopping, according to the identifier chosen with the mixture of Poisson distributions. Similarly,

[1] A function $f(t)$ is an infinitesimal of a higher order than t, namely $o(t)$, if $\lim_{t\to 0} \frac{f(t)}{t} = 0$.

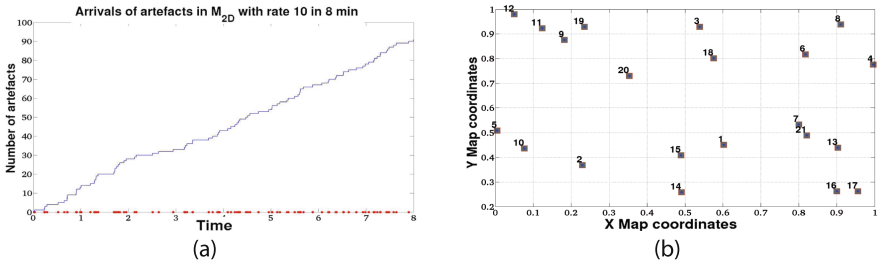


Fig. 3. (a) Simulation of the waiting times as well as of the trajectory of the Poisson process, regulating the arrivals of the artefact to M_{2D} (b) Simulation of the marked Poisson process, regulating the distribution in space of the artefacts, with the corresponding arrival time.

the motion planning algorithms Π take into account the current positions of the artefacts already in M_{2D} [31] and the gridmap to consistently move the object in the scene. Note that the finite-horizon Markov decision process $\mathcal{H}_A(t)$ selects the motion actions up to the time horizon d, according to the underlying action policy. Given the position of the other artefacts within M_{2D}, the motion planning Π plans collision-free short trajectories to reach the new position and orientation of the artefact, at time t', provided $\mathcal{H}_A(t)$ has selected a set of behaviour $b_k \in B$.

3.3 The AR-Builder Server

The AR-builder server interconnects the real environment model together with the simulation model of the artefacts. The AR-builder server relies on the `tf` software library. Namely, it keeps track, over time, of the transformations between the global reference frame $\mathcal{F}_E$, attached to the real environment model, the base frame $\mathcal{F}_R$, attached to the robot base, the base laser frame $\mathcal{F}_L$, attached to the laser sensor, and the camera frame $\mathcal{F}_C$, associated with the camera system, mounted on the robot. Given the `tf` library, the AR-builder server can determine the current pose of the reference frame $\mathcal{F}_A$ attached to the center of mass of the artefact and map it into the frame $\mathcal{F}_E$, rather than $\mathcal{F}_L$ or $\mathcal{F}_C$. Upon the registration of the artefact, the AR-builder server correctly places it within the real environment, by both projecting the bounding box b of the artefact on M_{2D} and concatenating the vertexes of the polygonal mesh $\mathcal{S}_A$ to the voxels of M_{3D} (see Fig. 4(a)). On the basis of these computational steps, the AR-builder server constructs the augmented model of the real environment (see Fig. 4(b)).

The AR-builder server implements a collision detection algorithm according to the dynamic model $\mathcal{H}_A(t)$. The algorithm, performing pairwise hit-testing, determines whether the bounding box of an artefact intersects the bounding box of another one or, alternatively, whether an artefact becomes embedded in the polygonal mesh $\mathcal{S}_E$ of the real environment. In such a case, the builder simply resolves the collisions by either moving back the artefact to its last known safe pose or by allowing the artefact to move up to a safe distance. The model of the

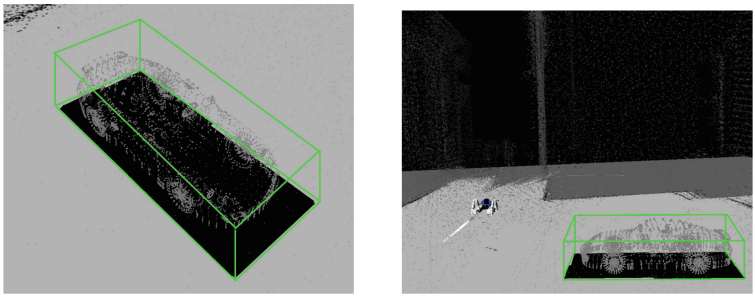


Fig. 4. (a) Projection of the bounding box b of the artefact on to M_{2D} (b) The 3D occupancy grid M_{3D} is augmented concatenating the vertexes of the polygonal mesh $\mathcal{S}_A$ associated with the artefact, to the voxels of M_{3D}

augmented real environment comprises only the artefacts which are not occluded, with respect to the robot field of view. These artefacts can be effectively ray-traced out from M_{2D}, thus not affecting the robot path-planning. The AR-builder server checks occlusion effects by implementing a ray tracing version of the z-buffer algorithm. Let $\mathcal{F}_{view} = \{P, (W, d_{max})\}$ be the view model of the real robot, where P is the camera matrix of the real robot with center C and W is the polyhedral cone of the field of view, such that $n(h^{(i)}) \times m(h^{(i)})$ is the dimension of the viewing plane at distance h_i, with h_{max} the maximum viewing ray length. We recall that the points X on the ray joining a point X and its projection on the image plane is $x = PX$, where X is actually a ray of points, that is why along this ray only the *free objects* are seen by the robots while the others are occluded. Finally the vector in the direction of the principal axis is defined by $det(M)m^3$, with $P = [M|p_4]$ [32]. To consistently deal with occlusions, an acclusion matrix is designed s follows. Let $d^0_{ij}, d^1_{ij}, \ldots, d^n_{ij}$ be the parameter values indicating where intersections with the implicit surface of the polygonal mesh $\mathcal{S}_{3D}$ of the real environment occur; let d^k_{ij} be the first positive root, for each pixel (i, j). We define a matrix $Z_{\mathcal{S}_{3D}} \in \mathbb{R}^2$ such that

$$z_{ij} = \begin{cases} d^k_{ij} & \text{if } d^k_{ij} \in [0, d_{max}]\,, \\ \infty & \text{otherwise} \end{cases} \tag{2}$$

Likewise, for each artefact a a matrix Z_a, mentioning the first positive roots $d^k_{ij} \in [0, d_{max}]$ is obtained by computing, for each pixel, the intersection of the corresponding ray with the bounding box b of the artefact. The set of artefacts perceived by the real robot is, therefore, given as follows:

$$\mathbf{A}_f = \{a | count((Z_a - Z_{\mathcal{S}_{3D}})_{ij} \leq 0) > \tau\}$$

Here the function $count(\cdot)$ returns the number of the elements (i, j) which satisfy the above condition, and τ is a threshold for assessing partial occlusions.

4 Experimental Results

In this section we illustrate the applicability of ARE to robot development and evaluation. The robotic platform is an UGV (see Fig. 5); two bogies on the sides are linked to a central body containing the electronics. Each bogie is made of a central track for locomotion and two active flippers on both ends to extend the climbing capabilities. A breakable passive differential system allows the rotation of the bogies around the body. Three sensors are installed on the platform; a rotating 2D laser scanner to acquire a 3D point cloud of the environment, an omni-directional camera for object detection and localization with a 360° field of view and an IMU/GPS for 3D localization.

A set of perception capabilities are embodied into the robot. The robot is provided with a real-time 2D and 3D ICP-based simultaneous localization and mapping (SLAM) system [33]. The robot is endowed with a path planning algorithm which generates short trajectories, enabling the robot to move within the environment, preventing the collision with the dynamic obstacles [34]. Finally a high level planner takes care of a mixed initiative control shared with the rescue operator [35,36].

We embedded the AR-based simulation framework into a ROS package. We deployed the robotic platform in a wide outdoor area, and set up two experiments, where ARE has been used to populate the real surroundings with artefacts.

In the first experiment, we wanted to check the robot ability to replan the path towards a goal location, as the frequency of the arrivals of the artefacts into the environment changes. Different parameter settings of the path-planner have also been settled, further affecting the robot behavior into the navigation task (see Fig. 6).

During the experiment the path-planner component computes a new path each time the scene is updated. To measure the robot ability to replan the following time ratio is introduced

$$\rho = \frac{\rho_t}{\rho_t + G_t} \tag{3}$$

(a) (b) (c)

Fig. 5. The robotic platform at work

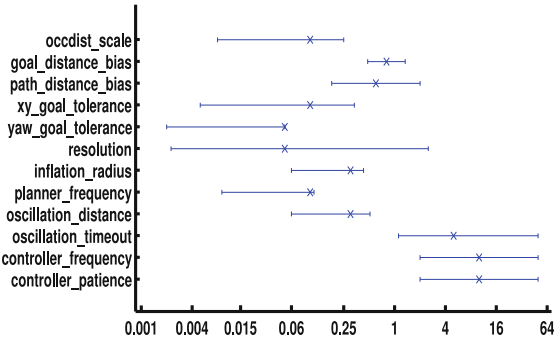
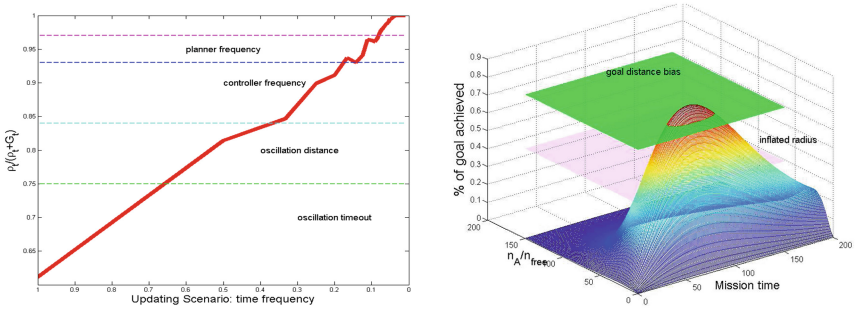


Fig. 6. Parameter names and range (note: x-scale is logarithmic)



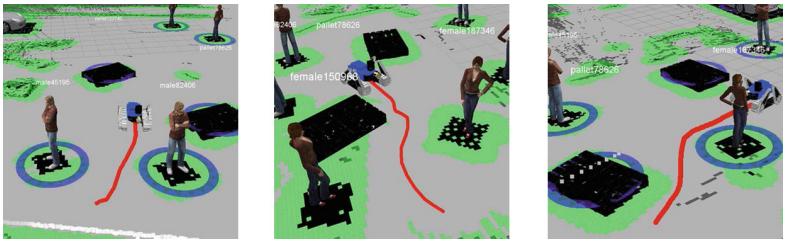
(a) (b)

Fig. 7. (a) The graph illustrates on the Y-axis the value of Eq. 3, against scene update frequency. This shows how the parameter settings of the path-planner affect, in terms of time, the ability of the robot to replan the path towards the goal location. (b) Here the graph illustrates the number of goal that can be achieved with increasing mission time, and given an increasing spatial complexity, with thresholds indicating the parameters limits. Mission time is specified in minutes, spatial complexity is an index, as indicated in Eq. (4).

Here ρ_t and g_t denote, respectively, the time needed to the path-planner to replan the path, and the estimated time to reach the goal location. Figure 7(a) shows how the time frequency at which the scenario is updated, with the arrivals of new artefacts, affects the replanning time, under different parameter settings of the path-planner, hence it affects the robot short-term navigation capabilities.

In the second experiment we tested the long-term capability of the robot to navigate the cluttered environment in order to reach several goal locations. In this experiment the space complexity of the environment, as well as the parameters of the path-planner related to the goal bias, have been taken into account. To measure the space complexity of the environment the following space ratio has been introduced:

$$\nu = \frac{n_{\mathbf{A}}}{n_{free}} \tag{4}$$



(a) (b) (c)

Fig. 8. Some snapshots of the experiment in progress, showing the augmented reality view (in ROS rviz), where is visible the 2D augmented costmap (black cells) with the inflated regions in green. The thick line in red is the path being executed by the path planner (Color figure online).

Here n_{free} and $n_{\mathbf{A}}$ denote respectively the number of free cells of the 2D occupancy grid $\mathcal{M}_{2D}$ of the mapped area and the number of the cells occupied by the set $\mathbf{A}$ of artefacts within the environment. The robot is instructed with the task to reach multiple goal locations. The path-planner computes the initial path to reach each goal and it replans a path from the robot current position to the current goal pose, whenever an artefact arrives into $\mathcal{M}_{2D}$, so as to find a collision free path, if one exists. Upon the receipt of the safe path, the execution component must be able to move the robot to effectively reach the current goal. In this experiment, the overall time needed to accomplish the task is measured together with the percentage rate of the reached goal locations. Figure 7(b) reports the performance of the robot in the navigation task with respect to different values of the space complexity of the environment (Fig. 8).

5 Conclusions

We propose a framework to augment the robot real world that advances the state of the art, as it introduces, together with the augmented environment, also the robot perceptual model of the augmented environment and the possibility of tuning the degree of confidence and uncertainty of the robot on what it is presented in the augmented scene. Besides being a compelling environment for robot programming, AR offers several tools for content authoring as well. Authoring AR tools can be classified according to their characteristics of programming and content design, in low and high level, considering the concepts abstraction and interfaces complexity incorporated in the tool. Programming tools are based on basic or advanced libraries involving computer vision, registration, three-dimensional rendering, sounds, input/output and other functions. ARTToolKit [37], MR [38], MX [39] and FLARToolKit are examples of low level programming tools. We presented the ARE framework together with a practical application of its use for robot parameter tuning. The experiments have shown how increasing complexity can affect planning and replanning abilities, and therefore that ARE is a promising experimental tool.

Acknowledgments. The research has been funded by EU-FP7 NIFTI Project, Contract No. 247870.

References

1. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.: Ros: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
2. Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., MacIntyre, B.: Recent advances in augmented reality. IEEE Comput. Graph. Appl. **21**(6), 34–47 (2001)
3. Neumann, U., Majoros, A.: Cognitive, performance, and systems issues for augmented reality applications in manufacturing and maintenance. In: Proceedings of IEEE Virtual Reality Annual International Symposium, pp. 4–11 (1998)
4. Molineros, J., Sharma, R.: Computer vision for guiding manual assembly. In: IEEE International Symposium on Assembly and Task Planning (2001)
5. Shekhar, R., Dandekar, O., Bhat, V., Philip, M., Lei, P., Godinez, C., Sutton, E., George, I., Kavic, S., Mezrich, R., Park, A.: Live augmented reality: a new visualization method for laparoscopic surgery using continuous volumetric computed tomography. Surg. Endosc. **24**, 1976–1985 (2010)
6. Milgram, P., Zhai, S., Drascic, D., Grodski, J.: Applications of augmented reality for human-robot communication. In: Proceedings of IROS, vol. 3, pp. 1467–1472 (1993)
7. Rastogi, A., Milgram, P.: Augmented telerobotic control: a visual interface for unstructured environments. In: Proceedings of the KBS/Robotics Conference (1995)
8. Amstutz, P., Fagg, A.: Real time visualization of robot state with mobile virtual reality. In: Proceedings of ICRA, vol. 1, pp. 241–247 (2002)
9. Brujic-Okretic, V., Guillemaut, J.Y., Hitchin, L., Michielen, M., Parker, G.: Remote vehicle manoeuvring using augmented reality. In: Proceedings of VIE, pp. 186–189 (2003)
10. Zaeh, M., Vogl, W.: Interactive laser-projection for programming industrial robots. In: Proceedings of ISMAR, pp. 125–128 (2006)
11. Bischoff, R., Kazi, A.: Perspectives on augmented reality based human-robot interaction with industrial robots. In: Proceedings of IROS, vol. 4, pp. 3226–3231 (2004)
12. Pettersen, T., Pretlove, J., Skourup, C., Engedal, T., Lkstad, T.: Augmented reality for programming industrial robots. In: Proceeding of IEEE/ACM International Symposium on Mixed and Augmented Reality, pp. 319–320 (2003)
13. Young, J., Sharlin, E., Boyd, J.: Implementing bubblegrams: the use of haar-like features for human-robot interaction. In: Proceedings of CASE, 298–303 (2006)
14. Dragone, M., Holz, T., O'Hare, G.: Using mixed reality agents as social interfaces for robots. In: The 16th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pp. 1161–1166 (2007)
15. Giesler, B., Salb, T., Steinhaus, P., Dillmann, R.: Using augmented reality to interact with an autonomous mobile platform. In: Proceedings of ICRA, vol. 1, pp. 1009–1014 (2004)
16. Stilman, M., Michel, P., Chestnutt, J., Nishiwaki, K., Kagami, S., Kuffner, J.: Augmented reality for robot development and experimentation. Technical report, Robotics Institute (2005)
17. Collett, T., MacDonald, B.: Augmented reality visualisation for player. In: Proceedings of ICRA, pp. 3954–3959 (2006)

18. Chong, J.W.S., Ong, S.K., Nee, A.Y.C., Youcef-Youmi, K.: Robot programming using augmented reality: an interactive method for planning collision-free paths. Robot. Comput. Integr. Manufactoring **25**(3), 689–701 (2009)
19. Green, S.A., Billinghurst, M., Chen, X., Chase, G.J.: Human-robot collaboration: a literature review and augmented reality approach in design. Int. J. Adv. Rob. Syst. **5**(1), 1–18 (2008)
20. Billinghurst, M., Kato, H., Poupyrev, I.: The magicbook: a transitional ar interface. Comput. Graph. **25**(5), 745–753 (2001)
21. Billinghurst, M., Belcher, D., Gupta, A., Kiyokawa, K.: Communication behaviors in co-located collaborative AR interfaces. J. HCI **16**(3), 395–423 (2003)
22. Moravec, H., Elfes, A.: High resolution maps from wide angle sonar. In: Proceedings of ICRA, vol. 2, pp. 116–121 (1985)
23. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
24. Wurm, K.M., Hornung, A., Bennewitz, M., Stachniss, C., Burgard, W.: Octomap: a probabilistic, flexible, and compact 3D map representation for robotic systems. In: Proceedings of ICRA Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation (2010)
25. Kamat, V.R., Dong, S.: Resolving incorrect visual occlusion in outdoor augmented reality using TOF camera and OpenGL frame buffer. In: Proceedings of NSF CMMI, pp. 1–8 (2011)
26. Finkenstadt, B., Held, L.: Statistical Methods for Spatio-Temporal Systems. Chapman & Hall/CRC, Boca Raton (2006)
27. Rathbun, S.L.: Asymptotic properties of the maximum likelihood estimator for spatio-temporal point processes. J. Stat. Plann. Infer. **51**(1), 55–74 (1996)
28. Vere-Jones, D.: Some models and procedures for space-time point processes. Environ. Ecol. Stat. **16**, 173–195 (2009)
29. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Stat. Soc.: Ser. B (Methodol.) **39**(1), 1–38 (1977)
30. Grimmett, G., Grimmett, G.: Probability and Random Processes, 3rd edn. Oxford University Press, Oxford (2001)
31. Müller, J., Stachniss, C., Arras, K.O., Burgard, W.: Socially inspired motion planning for mobile robots in populated environments. In: Cognitive Systems. Springer, Heidelberg (2010)
32. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2000)
33. Pomerleau, F., Colas, F., Siegwart, R., Magnenat, S.: Comparing ICP variants on real-world data sets. Auton. Robot. **34**(3), 133–148 (2013)
34. Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., Konolige, K.: The office marathon: robust navigation in an indoor office environment. In: Proceedings of ICRA, vol. 2010, pp. 300–307 (2010)
35. Carbone, A., Finzi, A., Orlandini, A., Pirri, F.: Model-based control architecture for attentive robots in rescue scenarios. Auton. Robot. **24**(1), 87–120 (2008)
36. Finzi, A., Pirri, F.: Representing flexible temporal behaviours in the situation calculus. In: Proceedings of the International Joint Conference of Artificial Intelligence IJCAI, pp. 436–441 (2005)
37. Kato, H., Billinghurst, M.: Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In: Proceedings of ACM International Workshop on Augmented Reality (1999)

38. Uchiyama, S., Takemoto, K., Satoh, K., Yamamoto, H., Tamura, H.: Mr platform: a basic body on which mixed reality applications are built. In: Proceedings of International Symposium on Mixed and Augmented Reality (2002)
39. Dias, J., Monteiro, L., Santos, P., Silvestre, R., Bastos, R.: Developing and authoring mixed reality with MX toolkit. In: IEEE International Augmented Reality Toolkit Workshop, 2003 (October 2003), pp. 18–26 (2003)