Robotics and Computer-Integrated Manufacturing 26 (2010) 240–249

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

A novel AR-based robot programming and path planning methodology

S.K. Ong [b,*], J.W.S. Chong [a], A.Y.C. Nee [a,b]

[a] Singapore-MIT Alliance, National University of Singapore, 9 Engineering Drive 1, Singapore 117576
[b] Department of Mechanical Engineering, National University of Singapore, 9 Engineering Drive 1, Singapore 117576

ARTICLE INFO

Article history:
Received 27 October 2009
Accepted 2 November 2009

Keywords:
Robot programming
Augmented reality
Reparameterization
End-effector orientation

ABSTRACT

This paper discusses the benefits of applying Augmented Reality (AR) to facilitate intuitive robot programming, and presents a novel methodology for planning collision-free paths for an *n* degree-of-freedom (DOF) manipulator in an unknown environment. The targeted applications are where the end-effector is constrained to move along a visible 3D path/curve, which position is unknown, at a particular orientation with respect to the path, such as arc welding and laser cutting. The methodology is interactive as the human is involved in obtaining the 3D data points of the desired curve to be followed through performing a number of demonstrations, defining the free space relevant to the task, and planning the orientations of the end-effector along the curve. A Piecewise Linear Parameterization (PLP) algorithm is used to parameterize the data points using an interactively generated piecewise linear approximation of the desired curve. A curve learning method based on Bayesian neural networks and reparameterization is used to learn and generate 3D parametric curves from the parameterized data points. Finally, the orientation of the end-effector along the learnt curve is planned with the aid of AR. Two case studies are presented and discussed.

© 2009 Elsevier Ltd. All rights reserved.

1. Introduction

Robots have been used to perform a wide range of tasks which would otherwise be time-consuming, strenuous and/or dangerous for the humans. There are currently three types of robot programming methods namely, lead-through, walk-through and offline programming [1]. These methods have a number of disadvantages. The lead-through method using a teaching pendent is slow and unintuitive. The walk-through method is only suitable for certain types of robots because the actuator brakes need to be disengaged. The use of lead-through and walk-through programming also poses safety concerns for the users. Hence, one motivation behind the use of offline programming is the safety issue.

Virtual Reality (VR) is an example of an offline method aimed at increasing the intuitiveness of the robot programming task for the human in a safe environment [2]. Natonek et al. [3] and Aleoti et al. [4] developed VR systems for the training of robots for object manipulation in environments that are known *a-priori*. However, desktop-based offline programming is less intuitive as compared to the lead-through and walk-through methods because the human is not participating in the working environment, and fully immersive VR environments are complex and costly. Offline programming is generally inflexible because of the need to model all the physical entities in the real environment. In addition, careful calibration and fine tuning are needed when a planned simulated environment is replicated in a real environment. Most offline programming packages also require the users to master complex programming languages that are non-generic and software specific. Hence, there is an increasing need to identify new ways to program robots safely, quickly and more intuitively.

In general, the robot programming task is time-consuming and unintuitive [5]. A simple task that can be performed easily by a human is often difficult to replicate using a robot. This is because humans possess the ability to perform complex manipulations without the need to consciously perceive the detailed motion plans [6]. Recently, an approach known as Programming by Demonstration (PbD) has emerged to address the intuitiveness in human–robot interactions. This approach is unique as the robot system *interprets*, and *improves* or *replicates* actions based on the data obtained from human demonstrations [7]. A PbD robotic system could potentially be as flexible as a human.

In this research, a robot programming methodology coupled with an Augmented Reality (AR) environment is proposed to address the above-mentioned issues. The target applications are non-contact tasks, e.g., arc welding and laser cutting, where the end-effector is required to follow a particular 3D curve at a consistent orientation with respect to the curve. The curve is visible to the user, however, its relationships with the robot/world coordinate systems and other objects in the environment, may

* Corresponding author. Tel.: +65 65162222; fax: +65 67791459.
E-mail address: mpeongsk@nus.edu.sg (S.K. Ong).

0736-5845/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.rcim.2009.11.003

not be known, i.e., a non-calibrated or unknown environment. The AR environment provides visual feedback that improves the interaction of the user with the real world during the path planning process. In the proposed methodology, PbD concepts are used where the user is responsible for performing a number of demonstrations of an unknown curve to acquire suitable input data for the system to learn the unknown curve, interactively defining the relevant free-space along the curve, and planning the orientation of the end-effector with respect to the curve. The primary goal is to enhance the flexibility, intuitiveness and efficiency of robot programming. The proposed methodology can be used in static and unknown environments, and utilizes the capabilities of the human in sensing and interpreting a new environment.

The rest of the paper is organised as follows: Section 2 briefly discusses the application of AR in facilitating interactive robot programming. Section 3 describes the setup of the proposed AR system. Section 4 outlines the proposed methodology for planning collision-free paths in an AR environment for the targeted applications. Section 5 presents and discusses the results of two case studies. Lastly, Section 6 concludes the paper and proposes future research opportunities.

2. Robot programming using AR

AR is an emerging technology that is derived from VR. It is an environment where computer-generated 2D or 3D objects are rendered onto a real world scene. An AR system involves two basic activities: (1) tracking, where the orientation and position of a camera relative to the entities in the real world scene are calculated and, (2) registration, where virtual objects are rendered onto the user's view based on the values calculated from the tracking process. 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 has been a subject of intensive research in many applications, such as maintenance [8], manual assembly [9] and computer-assisted surgery [10].

Relatively little work has been reported on the application of AR in robot programming. One of the first robotic applications using AR is in telerobotics control [11], where an operator is provided with visual feedback of a virtual wireframe robot superimposed over an actual physical robot located at its remote working environment. The operator will evaluate the tasks executed by the virtual robot. If it is satisfactory, the task will be transferred to a real robot. AR has also been used in the programming of painting robots [12], where a spray gun, which is handheld by a user, is tracked and the corresponding virtual spray is displayed to the user through a Head-Mounted Display (HMD). Feedback is obtained through tracking the object to be sprayed using markers, and the path travelled by the spray gun is then converted into a robot program.

An AR environment is suitable for developing interactive and intuitive robot programming, and complements the notion of PbD. This is because in an AR environment, the human is allowed to participate in the path planning process through demonstrating and performing certain tasks that are difficult to be simulated in a computer, such as recognizing and interpreting an environment. AR augments the view of a user with *visual guidance*. In the proposed approach, the user directly interacts with the virtual robot instead of being separated from it [11,12]. Various robotic options can be safely evaluated using different robot models without incurring significant costs. An *in-situ* approach using AR also does not require precise calibration between the various entities. Lastly, an AR environment can be used for programming large robots through the use of scaled-down models of the environment and virtual robots.

3. System implementation

The Robot Programming using AR (RPAR) system is implemented using the video-based tracking method in ARToolkit [13] and C programming language. The HMD consists of a single IEEE Firefly camera and an i-glass video display device. This setup is based on the inside-out tracking where markers attached to objects (static or moving) are tracked using a camera(s) attached to the HMD. The camera(s) provides the video images needed for image processing, and a view of the real world.

3.1. Relationships between coordinate systems

There are three main Euclidean coordinate systems in the RPAR system; the camera, the base of the robot, and the wrist of the robot, which is a probe held by a user to control and move the robot. As shown in Fig. 1, marker **B** is attached to the base and marker **A** to the wrist. The relationships between the camera (X_c, Y_c, Z_c), base (X_B, Y_B, Z_B) and wrist (X_W, Y_W, Z_W) coordinate systems are obtained using the marker detection method in ARToolkit, while the relationship between the wrist and the end-effector reference point ($X_{W'}$, $Y_{W'}$, $Z_{W'}$) coordinate systems is known. The end-effector reference point can also be the tip of the handheld probe used for general interaction. The matrix M relates any two coordinate systems and consists of the rotation, R and translation, T matrices. Therefore, the transformation from the base, B to the end-effector reference point, W' coordinate systems is given by

$$ {}^{B}_{W'}M = {}^{B}_{c}M[{}^{W}_{c}M]^{-1}\,{}^{W}_{W'}M = {}^{B}_{c}M\,{}^{c}_{W}M\,{}^{W}_{W'}M \tag{1} $$

3.2. Robot kinematics model

The robot configuration can be characterized by its generalized coordinates, $\mathbf{q}=[q_1, q_2, \ldots, q_n]$ where n is the number of degrees-of-freedom (DOF). To illustrate the methodology, a six DOF revolute joint robot (similar to the PUMA 560 industrial robot with three axes intersecting at the wrist [14]) is used. The angles of the joints are denoted as $\mathbf{q}=\boldsymbol{\theta}=[\theta_1, \theta_2, \ldots, \theta_6]$. Besides the PUMA 560-type robot described in this paper, other robot models can be used and this provides flexibility for the evaluation of various robotic options. The matrix ${}^{b}_{w'}M$ that relates the end-effector reference point to the base is the kinematics chain of the robot model that has to be solved.

The kinematics analysis can be divided into forward and inverse kinematics. For the latter, setting allowable joint ranges or using the geometric approach (only one arm configuration type is picked at a time) avoids multiple solutions and unwanted configurations, such as singularities where the robot loses a DOF. Avoiding multiple solutions at any one time is essential to

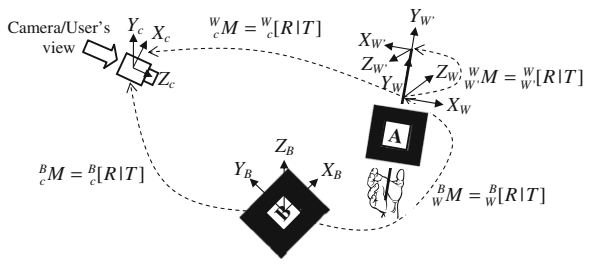

Fig. 1. Relationships between markers and camera.

Download English Version:

https://daneshyari.com/en/article/414581

Download Persian Version:

https://daneshyari.com/article/414581

Daneshyari.com