Robotics and Computer-Integrated Manufacturing 28 (2012) 227–237

Contents lists available at SciVerse ScienceDirect

Robotics and Computer-Integrated Manufacturing

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

Interactive robot trajectory planning and simulation using Augmented Reality

H.C. Fang, S.K. Ong *, A.Y.C. Nee

Mechanical Engineering Department, Faculty of Engineering, National University of Singapore, 9 Engineering Drive 1, Singapore 117576, Singapore

ARTICLE INFO

Article history:
Received 1 October 2010
Received in revised form
10 May 2011
Accepted 8 September 2011
Available online 29 September 2011

Keywords:
Augmented Reality
Robot programming
Human–robot interaction
Trajectory planning
Simulation

ABSTRACT

Human–robot interaction in industrial robotics has largely been confined to finding better ways to reconfigure or program the robots. In this paper, an Augmented Reality based (RPAR-II) system is proposed to facilitate robot programming and trajectory planning considering the dynamic constraints of the robots. Through the various simulation capabilities provided in the proposed AR environment, the users are able to preview the simulated motion, perceive any possible overshoot, and resolve discrepancies between the planned and simulated paths prior to the execution of a task. By performing the simulation, the performance of the trajectory planning and the fitness of the selection of the robot controller model/parameters in the robot programming process can be visually evaluated. Practical issues concerning the system implementation are also discussed.

1. Introduction

Recent developments and advancements in robotics have seen the emergence of various types of robots for a wide range of applications. These diverse applications have raised more challenges on the identification of intuitive interfaces for human–robot interaction (HRI) suitable for users who are not necessarily experienced in the field of robotics. There is an increasing demand in the development of robotics and automation solutions that are less expensive, safer, easier to install and re-program, and that can be adapted by small and medium-sized enterprises (SME) [1]. It would be desirable to have economically feasible solutions where the operator can work as an assistant with the complex robots to solve unpredictable problems [2]. Thus, an interface for more intuitive, quicker and safer robot programming would not only need to utilize the users/operators common cognition, but also to address the issues regarding the traditional programming methods for industrial robots.

Augmented Reality (AR) is the merging of virtual elements with the real world entities so that both can be perceived by the users at the same time. The use of AR in robotic systems enables the users to interact with the spatial environment through more intuitive interaction interfaces, such as a teaching stylus for manual guidance [3], and perceive instantaneous feedback through the integration of various types of sensors, such as force torque sensors [3], or optical tracking devices [4–6]. Numerous research efforts have been reported on AR for robots, and there are larger research initiatives, such as AVILUS [7,8] and SMErobot [3,9]. Many manufacturing companies, e.g., Siemens, Audi, BMW, etc., have also participated in various AR projects for product development, production and service [10].

In this paper, an approach for Robot Programming using Augmented Reality (RPAR-II) is proposed and presented. The main contribution of the approach is the incorporation of AR and robot dynamics in robot programming and trajectory planning and the transformation of the planned trajectories into task-optimized executable robot paths. A hand-held device, which is attached with a marker-cube, is used for human–robot interaction in the task and path planning processes. A time-optimal trajectory optimization scheme is adopted to obtain the approximated optimal trajectory profile rapidly, which is used to simulate the virtual robot with a suitable control scheme. In addition, this approach assists the users in the evaluation of the trajectory and provides cues and options to tune the controller parameters. The rest of the paper is organized as follows. Section 2 presents the related studies. Section 3 gives an overview of the proposed RPAR-II system. Section 4 introduces the HRI mechanism and outlines the proposed methodology for trajectory planning considering the effects of robot dynamics. Section 5 discusses the experimental results and practical implementation issues; and Section 6 gives the conclusion and future work.

2. Related studies

Unlike humanoids or companion robots which are equipped with high-level interfaces, such as tactile-based or vocal-based

* Corresponding author. Tel.: +65 65162222; fax: +65 67791459.
E-mail addresses: g0600952@nus.edu.sg (H.C. Fang), mpeongsk@nus.edu.sg (S.K. Ong), mpeneeyc@nus.edu.sg (A.Y.C. Nee).

0736-5845/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.rcim.2011.09.003

sensors, etc., industrial robots are usually provided with a teaching pendant and a PC-based offline programming interface connected to its controller. Thus, the process of programming industrial robots is usually unintuitive and/or time-consuming [1,11,12], and may even pose safety concerns, e.g., physical human–robot interaction (pHRI [13]).

Off-line robot programming methods have been developed to address the issues discussed above. Virtual Reality (VR) is one of such methods that has been proven to be useful in medical robots for surgeries [14] and tele-robotics for tele-operations [15,16], etc. Liu et al. [17] developed an off-line feature mapping algorithm for arc-welding robots in the cases when the workpieces or fixtures are redesigned or the robot workstation layout has been changed. However, the VR-based robot programming requires full *a priori* knowledge of the workpieces, working area and thus more computational resources, which is not suitable for semi-structured or loosely structured working conditions.

Programming by Demonstration (PbD) is another approach in robot programming that incorporates the users' expertise in demonstrating a task manually and leaving the robot to observe, follow or replicate the task in real-time [18]. However, the demonstrations may be suboptimal as a human may not be able to emulate the ability of a robot in terms of speed and accuracy when demonstrating the tasks [19–21]. Besides, it may be difficult for a robot to execute the demonstrated trajectory precisely due to the jerks and jitters that may occur during the demonstrations [22]. Recent research on robot programming is moving towards multimodal interfaces. Gestures, voices, etc., are used as high-level symbolic inputs to control industrial robotic systems [23–25]. Methods using these forms of inputs are intuitive and natural to the operators. However, these methods might not be robust in the industrial field, and can lead to possible improper or even incorrect task interpretations due to the presence of noise in the working environment.

AR has been successfully applied in the manufacturing field, like in maintenance [26], assembly [27], and manufacturing planning [28]. In robotics, it has been applied in various applications, such as tele-operations [25,29] and robotic surgeries [30]. An Augmented Reality cueing method was reported [31,32] to assist the users in navigating the End-Effector (EE) of a real robot using two joysticks. The visual cues allow the users to navigate the robot in a tele-operation task intuitively under display-control misalignment conditions. These studies showed positive effects of using AR on operator performance in *ad hoc* tele-robotic tasks.

Some works have been reported in the use of AR in robotic systems to address the robot programming issues. Chong et al. [4] presented a method to plan a collision-free path through guiding a virtual robot using a probe attached with a planar marker. Zaeh and Vogl [5] introduced a laser-projection-based approach where the operators can manually edit and modify the planned paths projected over the real workpiece through an interactive stylus. Reinhart et al. [6] adopted a similar HRI approach [5] in robotic remote laser welding (RLW) applications where the production cycle efficiency for the RLW process has been significantly improved with reduced set up and programming time.

The application of AR also provides the operators with various simulation options in robotic planning [33]. However, the present research mainly focuses on geometric path planning problems with the robot kinematics model. Chong et al. [4] adopted the beam search strategy to find an energy-optimal path through evaluating the amount of pseudo-energy needed by the joints to execute a pick-and-place task. Work has been presented in Refs. [5,6] where only the planned geometric paths that the robot needs to follow are projected on the real workpiece. The simulations were mainly conducted to test whether the planned path(s) are reachable and collision-free without considering the motion constraints (e.g., joint velocity and joint acceleration).

The generation of a time-scale trajectory for a given robotic task taking into consideration the effects of robot dynamics constraints has been a popular research topic. To reduce the computational complexity and resources required in solving this problem, the de-coupled approach [34], which decomposes the original problem into geometric path planning and trajectory tracking, has been well studied and reported. A number of optimization criteria have been reported, such as the minimum-time trajectory planning [35,36], minimum-jerk trajectory planning [37], and hybrid trajectory planning, where a tradeoff is made between the path duration and the mechanical energy of actuators [38,39], etc. Verscheure et al. [40] reformulated the time-optimal trajectory planning problem into a convex optimization problem and solved it using interior point methods [41]. These methods solve optimization problems with equality and inequality constraints by reducing them to a sequence of linear equality constrained problems, yielding good worst-case complexity by exploiting the problems' structure efficiently. In this research, a similar approach is implemented to obtain the time-scale trajectory of a robot after the geometric path has been generated given the starting point, goal point and a number of via-points of the robot path.

3. System overview

The setup of RPAR-II is shown in Fig. 1. It includes a SCORBOT-ER VII type manipulator arm, an electrical gripper, a robot controller, a desktop-PC, a desktop-based display, a stereo camera, and a hand-held device attached with a marker-cube.

3.1. RPAR-II AR environment

The augmented environment in RPAR-II consists of the physical entities that exist in the robot operation space, such as the robot manipulator, the working platform, tools, workpieces, etc., and a virtual robot model which includes the virtual EE to replicate the real robot.

In this research, a virtual SCORBOT-ER VII manipulator is modeled in SolidWorks®. The geometries of the coupled joints and links are modeled separately according to the kinematic parameters of the manipulator. The links are built with both functional and geometric features so as to enhance the visualization effects. Separate components are exported in the *.STL* format,

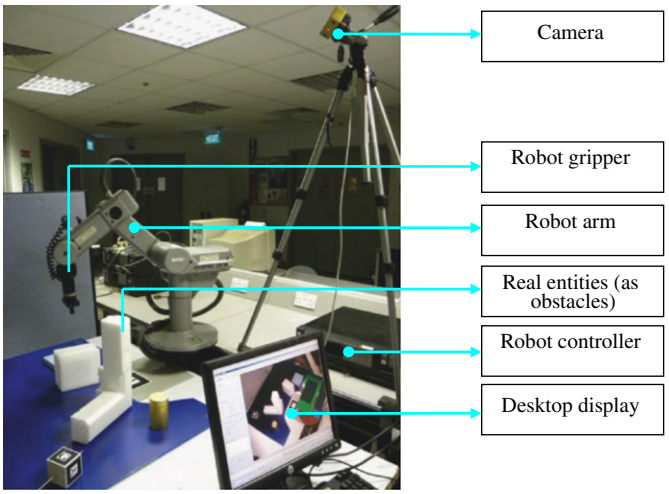

Fig. 1. The RPAR-II setup.

which contains the geometrical information with respect to the corresponding coordinate frame assigned to each link. The virtual robot model can be assembled and augmented on the real environment, and can be scaled up or down for different robot configurations.

3.1.1. Robot kinematics model

The kinematics configuration of a robot is characterized by its generalized coordinates $\mathbf{q}=[q_1,q_2,\ldots,q_n]$, where n is the number of degrees-of-freedom (DOF). For the SCORBOT-ERVII manipulator, which has five revolute joints, the joint angles are denoted as $\mathbf{q}=[\theta_1,\theta_2,\ldots,\theta_5]$. The path of the EE of the robot is achieved by solving the inverse and forward kinematics. The forward kinematics analysis is straightforward as any configuration of the robot in a joint space can be uniquely mapped to a configuration in the operation space. The classical kinematics solver is based on Denavit–Hartenberg (D–H) convention [34]. The inverse kinematics can be solved analytically given the position and orientation of the EE with respect to the base of the robot. Setting the joint ranges enables the kinematics solver to eliminate some of the unwanted configurations caused by joint redundancy, such as the singular configurations at the boundary of the operation space. The EE of the robot can be kept at the last feasible configuration prior to the operation space violation [4]. It is intuitive for a user when moving the EE of the robot along a given Cartesian path to verify whether all the points on this path are reachable, and/or whether its start point and goal point are reachable in the different solutions. These are the two essential geometric problems with Cartesian paths [34].

3.1.2. Robot dynamics model

The dynamic model of a robot is useful for the simulation of the robotic system; various robotic tasks can be examined without the need of a real robotic system. A valid model that represents the kinematics and dynamics properties of a robot will help the users learn and understand the mutual relationships between the torques applied to each joint and the resulting motion of the manipulator. Moreover, the predicted motion of the given manipulator under different given initial conditions, applied torques, etc., can be simulated by utilizing the robot dynamics model.

In this research, recursive Newton–Euler equations are adopted to compute the forward and inverse robot dynamics [42]. A user interface is implemented by which a user can easily load or modify the robot kinematics and dynamics parameters in terms of each link of a serial manipulator, such as the D–H parameters, link mass, center of gravity (CG), inertia tensor, motor properties, etc., during robot trajectory planning. This allows the users to evaluate the planned performance based on different robot parameters.

3.1.3. Virtual robot model registration

In the RPAR-II system, a vision-based method is adopted to register a virtual robot over the real robot, as shown in Fig. 2. The camera is maintained in a fixed position and orientation with respect to the workspace, such that the robot registration is only required to be performed once. Camera calibration is carried out to obtain the intrinsic parameters of the camera before it is used in the system. To enhance the visualization of the complete robot programming process, the camera is placed such that it could cover the operating space of the robot, or at least the required workspace for a given task.

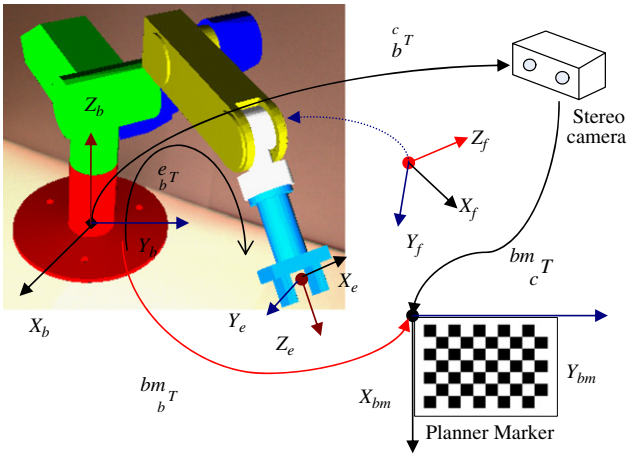

Fig. 2. Relationship between different coordinate systems.

3.2. System architecture

Fig. 3 presents the architecture of the RPAR-II system. In this system, an interaction device is used to guide the virtual robot to facilitate the demonstration of a path for a given task. Different routines are needed to perform the demonstrations and interactions for different types of tasks. Given a task consisting of pick-and-place operations, such as materials handling, etc., a Collision-Free Volume (CFV) will be first constructed through human demonstration(s). Next, given the starting and goal points of the task, via-points selection and possible via-points insertion and/or deletion will be performed. The interpolation between these points gives the geometric path of the task. For demonstration of a path-tracking operation, such as arc welding, etc., a user guides the EE of the virtual robot model following the desired path. The data sampled and gathered from the demonstrations can be parameterized and learned as an approximation of the desired path.

Once the geometric path has been obtained, the trajectory planning process proceeds with robot kinematics and dynamics constraints, such as the joint angle, velocity, acceleration limits, joint torque limits, etc., and user-defined requirements/criteria, such as minimum time, minimum energy consumption, etc., to complete the task. Prior to translating the time-scale trajectory into the robot controller codes, a simulation of this path under a given control scheme will be carried out using the virtual robot. The planned path and simulated path can be overlaid on the real workspace simultaneously, where the performance of trajectory planning and the fitness of the selection of the robot controller model/parameters can be evaluated visually (qualitatively).

4. The RPAR-II methodology for trajectory planning

Most of the tasks executed by the industrial robots can be decomposed into either one or a combination of (1) pick-and-place operation and (2) path-following operation. The focus of this research is on the planning of robotic pick-and-place operations, and the goal is to generate collision-free paths and time-scale trajectories that can be transferred directly into the robot controller codes, taking into account robot dynamics constraints and user-defined/task-dependent requirements. Instead of searching for a collision-free path directly between the predefined starting point and goal point [4], the proposed approach allows the users to create a series of via-points interactively within the CFV to form a path. There are two reasons for doing this. First, the search process, which is conducted in the joint space, cannot guarantee that the

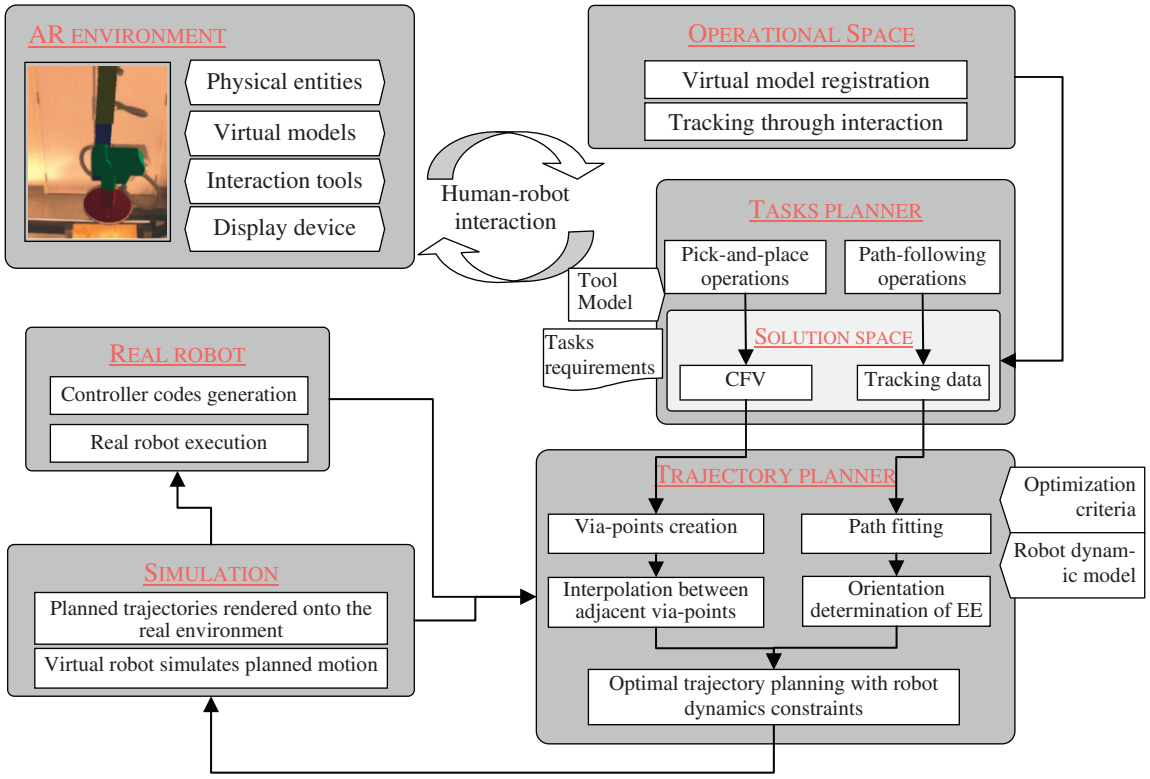

Fig. 3. Architecture of the RPAR-II system.

resulting path will be smooth in the Cartesian space. Second, searching for a time-scale trajectory directly within the CFV subject to both robot kinematics and dynamics constraints is a computationally intensive and complex problem deemed difficult to solve and implement. The path is formed through interpolation using the via-points created, and this process serves as the first step of the de-coupled approach for robot motion planning.

After mapping this path to the joint space, an optimal time-scale trajectory is obtained by solving the reformulated trajectory optimization problem subject to robot kinematics and dynamics constraints. Finally, the trajectory is fed to simulate the virtual robot performing the trajectory in the real environment, for the user to visually evaluate the quality of the resulting trajectory. The trajectory finally can be transferred into the controller codes which can be executed on a real robot. The implementation of the RPAR-II approach is illustrated in Fig. 4.

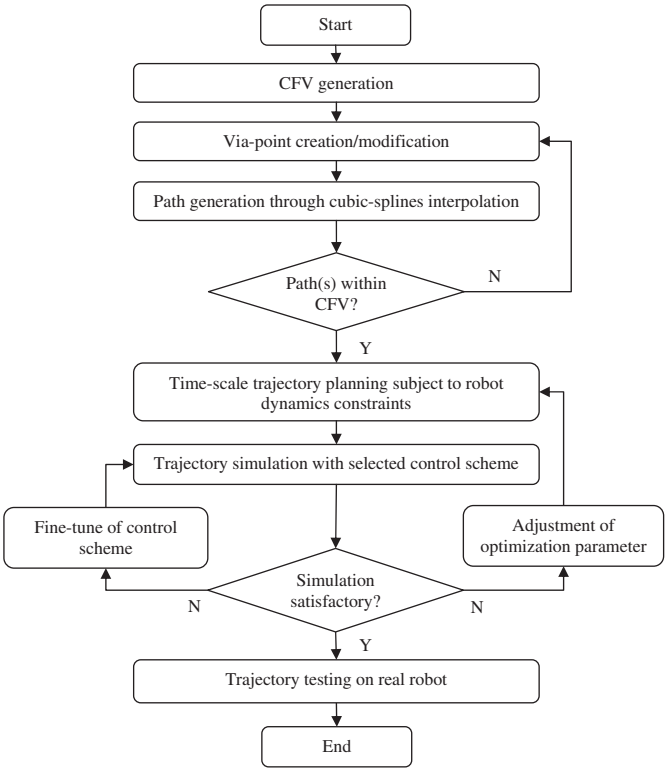

Fig. 4. Trajectory planning using the RPAR-II system.

4.1. Human–virtual robot interaction

4.1.1. Marker-cube

The use of a marker-cube to facilitate robot planning has been reported by Bischoff and Kurth [33]. In this research, a marker-cube attached on a probe is adopted as the human–virtual robot interaction device. Four different planar markers are attached to the faces of a cube. The advantage of using multiple markers over a single marker is that the cube can be tracked even if it has undertaken a relatively large rotation, providing more flexibility in guiding the virtual robot. However, multiple markers on a cube can introduce jitters due to the alternation of the recognized markers. To reduce the presence of jitters and enable the cube to be tracked continuously, calibration for the marker-cube is required. The procedure can be summarized as follows: (1) select a dominant marker and define the coordinate system of the marker-cube with reference to this marker; and (2) obtain the transformation from a recognized 2D marker to the dominant marker so as to obtain the coordinate system of the cube.

In the path planning process, the human operator is involved in (1) CFV generation; (2) via-points creation and (3) via-point

modification, which includes insertion of new via-point(s) and/or deletion of unwanted via-point(s).

4.1.2. CFV generation

In the trajectory planning process, a virtual sphere with a pre-defined radius is attached to the marker-cube and a CFV can be generated by recording the position of the center of the sphere through tracking the marker-cube. A CFV can be represented by a set of $\boldsymbol{N_0}$ virtual spheres in Eq. (1)

$$CFV = \bigcup_{i=1}^{N_0} \{X_i : S(c_i,X_i) \le r_0\} \tag{1}$$

c_i is the center of the sphere, r_0 is the pre-defined radius, and $S(c_i,X_i)$ is the Euclidean distance between c_i and X_i. The coordinate system of the base of the robot is used as the world coordinate system.

It is assumed that the virtual object is rigidly attached to the EE. A bounding cylinder is adopted as a swept model in the RPAR-II system. This swept model encloses both the EE and the virtual object and is used to facilitate the CFV check, which determines whether the virtual object and the EE are within the CFV and hence collision-free.

4.1.3. Via-points creation

The via-points created in the Cartesian space should meet two requirements: (1) they are reachable by the EE of the robot; and (2) at each via-point, the swept model should be within the CFV. In addition, the via-points should be selected at the locations that are adjacent to the critical zone of the volume (i.e., with higher curvature) as shown in Fig. 5. This increases the chance that the final interpolation between the created points is within the CFV. It is obvious that the via-points selection option in Fig. 5(b) is better than that in (a).

In the Cartesian space, a cylinder (BC) can be defined by four parameters, namely, the radius (r), height (h), origin (o) and axis (z) of the cylinder, as given in Eq. (2)

$$BC = BC(o,z,r,h) \in \Re^3 \tag{2}$$

The bounding cylinders corresponding to the created via-points can be defined by Eq. (3)

$$BC = \{BC_j(o_j,z_j,r,h,t_j) \in \Re^3;\ j = 1,\ldots,N_p\} \tag{3}$$

r and h are the radius and height which are constant for a given bounding cylinder; o_j and z_j are the origin and axis of the cylinder corresponding to the jth via-point; N_p is the number of via-points; and t_j indicates the sequence of the via-points to be created.

4.1.4. Via-point modification—insertion and deletion

Once a set of via-points has been created, a path that passes through all these points can be automatically generated through interpolation. If the path collides with the CFV, the via-point(s) can be modified. In practice, it is not necessary to re-generate a CFV and re-create all the via-points, as the quality of a CFV is task-independent and is affected by the jitters and incoherencies in human demonstrations as well as the frame rate achievable in the proposed system [4]. There are other occasions that require via-point modification, e.g., deletion of an inadequate point which is accidentally created by the user, or the insertion of a new via-point as in the case illustrated in Fig. 5(a).

It may be difficult for a user to locate a via-point in the workspace using the probe attached with a marker-cube. Therefore, a Euclidean distance-based approach is proposed to assist the user in selecting the point of interest in via-point deletion or insertion operations. This method computes the distances between the probe and each of the via-points, and associates each value with the corresponding via-point. The distances computed are updated automatically when the probe is moving in the workspace during via-point modification process. At each time instance, the via-point that has the minimum distance to the probe is nominated as the candidate point being selected, and is highlighted to differentiate from the rest. Defining the origin of the coordinate system of the interactive device (tip of the probe) $O_0(x_0,y_0,z_0)$, the ith via-point $V_i(x_i,y_i,z_i)$, and the Euclidean distance between O_0 and V_i, $\overrightarrow{O_0V_i}$, for $i=1,2,\ldots,N_p$, where N_p refers to the number of via-points with respect to the world coordinate system, the procedures for modification of via-point are shown in Fig. 6, which includes deletion of a via-point V_j (Fig. 6(a)) and insertion of a point between the via-points V_j and V_{j+1} (Fig. 6(b)).

4.2. Path interpolation

One of the objectives for path planning in the Cartesian space is to minimize the Euclidean distance between the starting point and goal point. In this research, given a set of via points, a cubic-spline interpolation is adopted to generate a piecewise polynomial path that is second-order continuous at the via-points [34]. Particularly, a set of cubic splines are formed with "clamped" conditions, which means only the first derivatives of the splines at the starting point and goal point are constrained, such that the motion of the robot could start and end with a user-defined velocity.

The path interpolated in the Cartesian space can be mapped to the joint space through inverse kinematics techniques [34]. As the via-points are created through guiding a full-scaled virtual robot in the robot's operation space, if all the cubic-spline segments are within the CFV, the resulting path will be reachable by the EE of

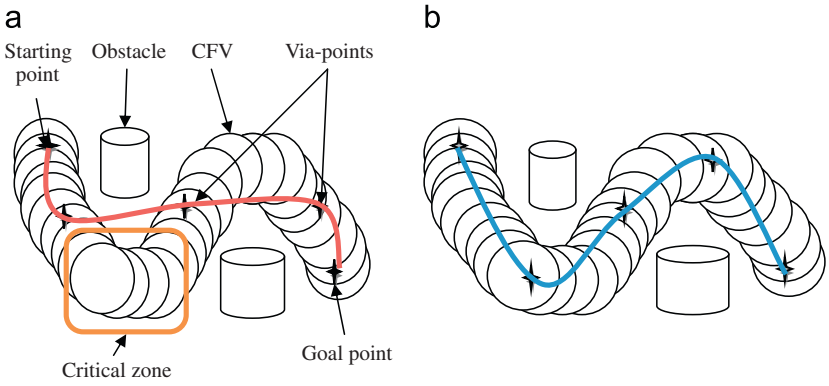

Fig. 5. Selection of via-points within the CFV during path planning.

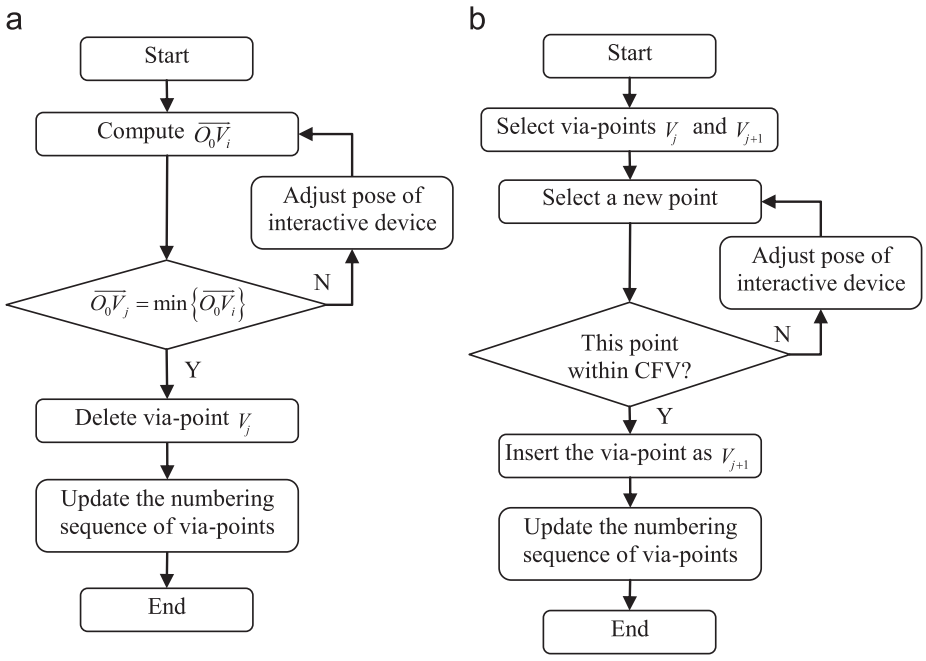

Fig. 6. Via-point modification procedure. (a) Via-point deletion procedure and (b) via-point insertion procedure.

the robot without violating the joint limits. Nonetheless, two practical issues would need to be considered, namely, the number of via-points and the data spacing among the via-points.

4.2.1. Number of via-points

The number of via-points is dependent on the task as well as the working environment, and to some extent affects the overall curvature of the resulting path. In the proposed approach, a small number of via-points will produce a path where one or more segments are likely to be outside the CFV. This problem can be solved by inserting one or more new via-points. However, the number of via-points should not be more than necessary. There are two reasons. First, the path interpolated will not become smoother and/or with smaller curvature through increasing the number of via-points. Second, due to the variations and uncertainties introduced during via-point creation/modification and the level of accuracy achievable by the tracking system, increasing the number of via-points means increasing the overall curvature of the path to be generated. In a less obstacle-clustered working environment, a feasible way for the user is to first select 5–10 via-points to perform the interpolation. If the generated path is not satisfactory, the user can progressively insert new via-points.

4.2.2. Time stamps attached to via-points

According to the creation procedure presented in Section 4.1.3, each of the via-points is characterized by its sampling sequence. However, it would not be practical to convert the sampling sequence directly to time stamps with equal data spacing used for path interpolation. There are two reasons. First, it is unlikely and not possible for a user to select a sequence of via-points such that they are spaced evenly in the interpolated path. Second, it is not even necessary to create the points with equal data spacing, as there should be more in the obstacle-intensive areas and fewer in less obstacle-crowded areas. Considering the Euclidean distance between two adjacent via-points, the normalized time stamp assigned to each via-point is proportional to the cumulative distance from the starting point to this point. The time stamp for the jth via-point is given in Eq. (4)

$$t_j = \frac{\sum_{i=1}^{j} d_i}{\sum_{i=1}^{N+1} d_i}, \quad j = 1,2,\ldots,N \tag{4}$$

d_i is the Euclidean distance between the jth via-point and the previous point, and N is the total number of via-points (except starting point and goal point, which are defined by $t_0=0$ and $t_{N+1}=1$, respectively).

The plots in Fig. 7 show the influence of the data spacing on the interpolation results. The dashed lines in Fig. 7(c) represent the plots corresponding to the interpolation with unequal data spacing. It can be observed that the paths interpolated with unequal data spacing are smoother in the Cartesian space (Fig. 7(b)) and have better distribution in terms of curvature in the joint space (Fig. 7(c)).

4.3. Trajectory planning

Trajectory planning involves finding a sequence $q=[q(0),\ldots,q(T)]^T$ that fulfills the task requirements and certain optimization criteria without violating the robot kinematics and dynamics constraints. In this research, a time-optimal trajectory is obtained through solving a convex optimization problem. The path duration is reformulated to be the optimization criterion, as has been described in Ref. [40]. The joint torque constraints are reinforced as a log-barrier item to the convex optimization problem, which allows the robotic system to maximize its productivity without having to engage its actuators excessively.

4.3.1. Time-optimal trajectory planning

Let the trajectory sequence given by a scalar path coordinate $\boldsymbol{s}$, which is usually interpreted as the arc length be $q=[q(s_0),\ldots,q(s_T)]^T$. Its first-order derivative and second-order derivative are given by $\dot{q}(s) = q'(s)\dot{s}$ and $\ddot{q}(s) = q''(s)\dot{s}^2 + q'(s)\ddot{s}$, where $\dot{s} = \frac{ds}{dt}$; $\ddot{s} = \frac{d^2s}{dt^2}$.

Assume the trajectory starts at time 0 and ends at time T, the path coordinate $\boldsymbol{s}$ is parameterized as $0=s_0=s(0)<s(t)<s(T)=s_T=1$, a

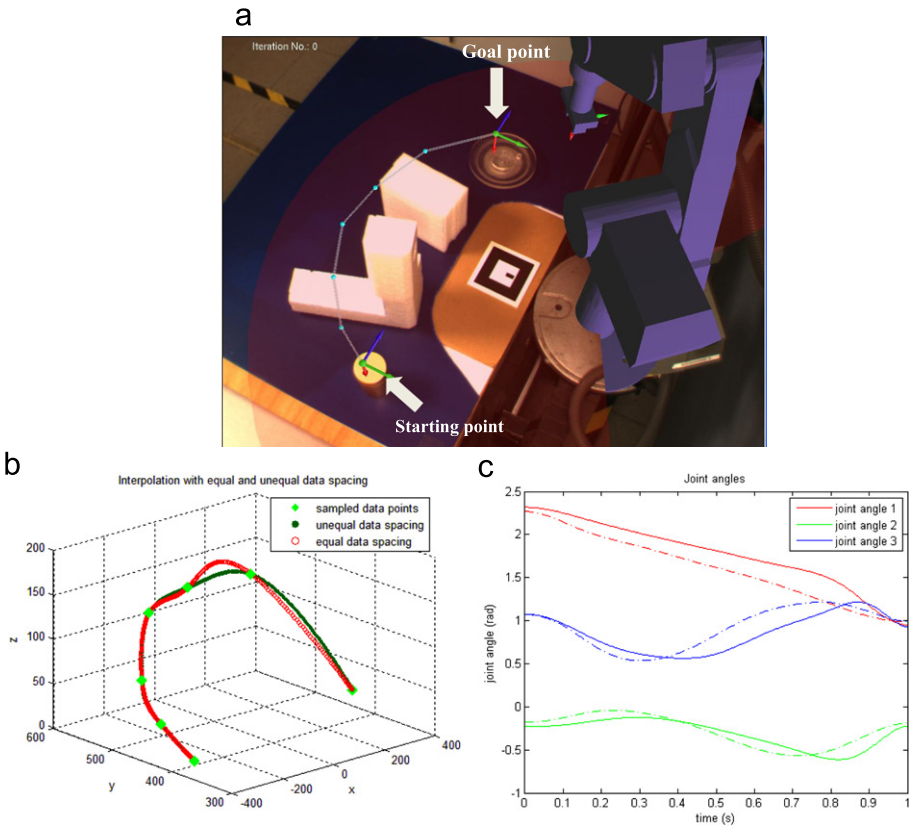

Fig. 7. Path interpolation from via-point set with equal and unequal data spacing. (a) Starting point, goal point and via-points in camera's view; (b) plots of the interpolated paths in the Cartesian space; and (c) plots of the paths in the Joint space.

strictly monotone representation where $t\in(0,T)$; $\boldsymbol{s}$ is discretized into N_s segments evenly with N_s+1 grid points. $\dot{s}$ and $\ddot{s}$ are the pseudo-velocity and pseudo acceleration of the trajectory, respectively. Defining $a(s)=\ddot{s}$ and $b(s)=\dot{s}^2$ as two optimization variables, the time-optimal trajectory planning is to minimize the path duration T subject to the robot dynamics constraints, as being represented in Eqs. (5)–(7)

$$E_T=\min T=\min\int_0^1\frac{1}{\dot{s}}ds\approx\min_b\sum_{i=0}^{N_s-1}\left(\int_{s^i}^{s^{i+1}}\frac{1}{\sqrt{b(s)}}ds\right) \tag{5}$$

subject to

$$\begin{aligned}&\underline{\tau}(s)\le\tau(s)\le\overline{\tau}(s)\\&b(s)\ge 0;\quad b'(s)=2a(s)\\&b(0)=\dot{s}_0;\quad b(1)=\dot{s}_T\\&0\le s\le 1\end{aligned} \tag{6}$$

where

$$\begin{aligned}&\tau(s)=m(s)a(s)+c(s)b(s)+g(s)\\&m(s)=M(q(s))q'(s)\\&c(s)=M(q(s))q''(s)+c(q(s),q'(s))q'(s)\\&g(s)=F_s(q(s))\mathrm{sgn}(q'(s))+G(q(s))\end{aligned} \tag{7}$$

$m(s)$, $c(s)$ and $g(s)$ are evaluated from the equation of motion for a manipulator [34]. Based on the assumption that function $b(s)$, which is associated with the pseudo-velocity of the path, is at least piecewise linear at each segment, Eq. (5) can be reformulated into a convex optimization problem given in Eq. (8) [39]

$$E_T=\min_b\sum_{i=0}^{N_s-1}\left(\frac{2\Delta s^i}{\sqrt{b^{i+1}}+\sqrt{b^i}}\right) \tag{8}$$

4.3.2. Log-barrier method

In robot trajectory planning, a dominant constraint in robot dynamics is the joint torque limits, i.e., the joint torque that produces the planned path should not exceed the actuator capability. Chan and Dubey [43] proposed a scheme to avoid the joint limits during trajectory planning. By substituting the joint limits and the planned joint angle with the joint torque limits and the planned joint torques, respectively, a penalty function $\boldsymbol{P}$ to avoid the joint torque limits is given by an averaging sum-log function in Eq. (9)

$$P=\frac{1}{2*ndof}\sum_{j=1}^{ndof}\left(\log\left(\frac{(\overline{\tau}_j-\underline{\tau}_j)^2}{4\times(\overline{\tau}_j-\tau_j)\times(\tau_j-\underline{\tau}_j)}\right)\right) \tag{9}$$

$\overline{\tau}_j$ and $\underline{\tau}_j$ represent the upper and lower bounds of joint torque, *ndof* is the DOF of the manipulator. This function gives higher penalty automatically when the computed torque is close to the torque bounds and the function value goes to infinity when the torque limits are violated. If the joint torque limits are symmetric, i.e. $\overline{\tau}_j=-\underline{\tau}_j$, Eq. (9) can be simplified as Eq. (10)

$$P=\frac{1}{2*ndof}\sum_{j=1}^{ndof}\left(\log\left(\frac{(\overline{\tau}_j)^2}{(\overline{\tau}^2{}_j-\tau_j^2)}\right)\right) \tag{10}$$

By augmenting Eq. (8) with the penalty function (10), which is known as the log-barrier item, the time-optimal trajectory planning problem is approximated as an unconstrained optimization problem [41]. The formulation is given in Eq. (11)

$$E_{T-\Gamma}=\min\sum_{i=0}^{N_s-1}\left(\frac{2\Delta s^i}{\sqrt{b^{i+1}}+\sqrt{b^i}}+\frac{\kappa}{N_s}P_i\right) \tag{11}$$

P_i is the penalty, as given in Eq. (9) or (10), evaluated at the middle point of segment $[s^i,s^{i+1}]$; κ is the log-barrier parameter.

This formulation is convex, which means it can be solved efficiently with a user-defined threshold within a number of iterations that is bounded by a polynomial of the problem dimensions (i.e., the number of objective and constrained functions). In this research the solution of the formulation is obtained through solving first-order optimality conditions of this equation. The log-barrier parameter κ is interpreted as the maximum gap between the solutions to the log-barrier approximated problem given in Eq. (11) and the time-optimal trajectory planning problem given in Eq. (8). With a small value of parameter κ (such as $\kappa=0.01$), the log-barrier item has negligible influence on the solution to the problem in Eq. (8). In this case the resulting joint torque will lead to "bang-bang" actuator behavior, i.e., at least one of the actuators is saturated. On the contrary a larger κ will produce a sub-optimal yet smoother solution at the expense of moderately longer path duration. This is evident from the fact that the path duration is not proportional to the pseudo-velocity $t=\sum_{i=0}^{N-1}\left((2\Delta s^i)\Big/\left(\sqrt{b^i(s)}+\sqrt{b^{i+1}(s)}\right)\right)$. With the aforementioned property, the log-barrier parameter can be used to tune the optimization process such that it yields a more practical solution to implement, in the sense of less aggressive usage of the actuators, on a real robotic manipulator.

To solve the unconstrained optimization problem given in Eq. (11), the optimization variables need to be initialized to fulfill constraints in Eqs. (6) and (7). Since $\mathbf{b}$ is at least twice differentiable and is known at the starting and end points, $\mathbf{b}^0$ can be defined as a parabola with a scalar as in Eq. (12)

$$\mathbf{b}^0=(\dot{s}_T-\dot{s}_0-c)s^2+c\cdot s+\dot{s}_0 \tag{12}$$

$\dot{s}_0$ and $\dot{s}_T$ are the pseudo-velocities at the starting and goal points, which could be zero or non-zero user-defined constants. The coefficient c is firstly chosen to be a positive value such that $\dot{s}_T-\dot{s}_0-c<0$, and then progressively decreased (while keeping $\dot{s}_T-\dot{s}_0-c<0$) until the joint torques constraints are satisfied.

4.4. Trajectory simulation

In the proposed system, a PD control scheme is adopted to simulate the virtual robot performing this trajectory. A set of PD control gains has been defined initially and an interface is developed allowing the control gains to be adjusted manually. During simulation, to check whether the joint torque constraints are violated, the normalized torque of each joint is computed and compared along the trajectory. The normalized torque for joint i is defined as Eq. (13)

$$\tau_u(i)=\left|\frac{\tau(i)}{\tau_{MAX}(i)}\right| \tag{13}$$

$\tau(i)$ is the actual joint torque of joint i computed according to the selected control gains, and $\tau_{MAX}(i)$ is the joint torque bound for joint i.

The normalized joint torques are evaluated at each sample time during simulation. According to Eq. (13), if $\tau_u(i)\in(0,1]$, which means the computed joint torque satisfies the joint torque constraints, the link that is associated with the joint with the largest $\tau(i)$ among all the joints is highlighted to indicate that it is the link most likely to deviate from the planned trajectory. Alternatively, if $\tau_u(i)>1$, the simulation will be paused and the virtual robot will be halted at the current pose. The user can tune the control gain associated with joint i, then run the simulation again from the starting point. There are some cues that roughly tell which parameter should be tuned. For instance, the planned trajectory is approximately time-optimal, which means the joint velocity is relatively high, hence the derivative control gains may affect the output of the control system the most. If the simulated trajectory is found to be shifted away parallel to the planned trajectory, then the proportional control gains need to be adjusted as these parameters dominate the drift of the output of the control system. In general, a number of iterations are required to reduce the deviation between the planned and simulated trajectories to a predefined threshold.

5. Implementation and discussions

This section presents a case study on the proposed methodologies which assist a user in planning a collision-free trajectory for a pick-and-place task subject to robot dynamics constraints. The RPAR-II was implemented using C/C++ programming language under Visual C++ 2005 environment on a 1 GHz PC. Two external packages are used, namely, *Roboop* [42] which provides robot kinematics and dynamics modeling, and *gnuplot* [44] which provides various plot routines.

- RPAR-II system GUI
  A Graphic-User Interface (GUI) was developed to assist the users in robot task and path planning using a virtual robot in the real working environment. Two panels were implemented to perform different functionalities at different stages of the planning process: one for virtual robot modeling and another for trajectory planning and simulation. The panel for virtual robot modeling enables the users to create parametric models manually for different robot types by cooperating with the *Roboop* package. Another panel offers a number of options for trajectory planning and simulation, including the selection of different log-barrier parameters, the different plot options for planning results analysis, such as the path duration and the energy consumption of actuators, and the tuning options for control gains adjustment.
- Geometric path generation
  Fig. 8 illustrates the process of planning a collision-free path for a robotic task, which is to transfer an object from the starting point to the goal point. In this example, a total of six via-points (excluding the starting and goal points) are finally created, and each is assigned with normalized spaced time stamp.
  Fig. 9 shows the procedure to modify the list of via-points in the case where the resulting path generated from these points is outside the CFV. As only three via-points have been created (Fig. 9(a)), thus one possible way of solving this problem is to insert a new via-point that is likely to "drag" this path segment back into the CFV, as shown in Fig. 9(b), (c). Fig. 9(d) shows a path re-generated from the updated via-point sequence, and it can be observed that the path has better properties in terms of curvature distribution and smoothness. Nevertheless, if there are more via-points than necessary have been created, which might lead to an over-curved path, one or more undesired via-points can be deleted by following the procedure presented in Fig. 6(a).
- Trajectory planning
  In this case study, the path coordinate $\boldsymbol{s}$ is unified, i.e. $s\in[0,1]$, and the step length is set as 0.005 ($N_s=200$). Eq. (7) is evaluated at each step along the path to form the coefficients of equations representing the joint torques constraints. An estimation of kinematics and dynamics parameters of SCORBOT-ER VII model is adopted from Ref. [34]. The choices of the log-barrier parameter are logarithmically spaced, i.e., 0.01, 0.04, 0.158, 0.631 and 2.51. In this case study, the parameter has been chosen to be 0.631.The optimization variables $\mathbf{b}^0$ are initialized using Eq. (12) where the pseudo-velocities at the starting and goal points are zero, given an initial value of the constant $c=100$.

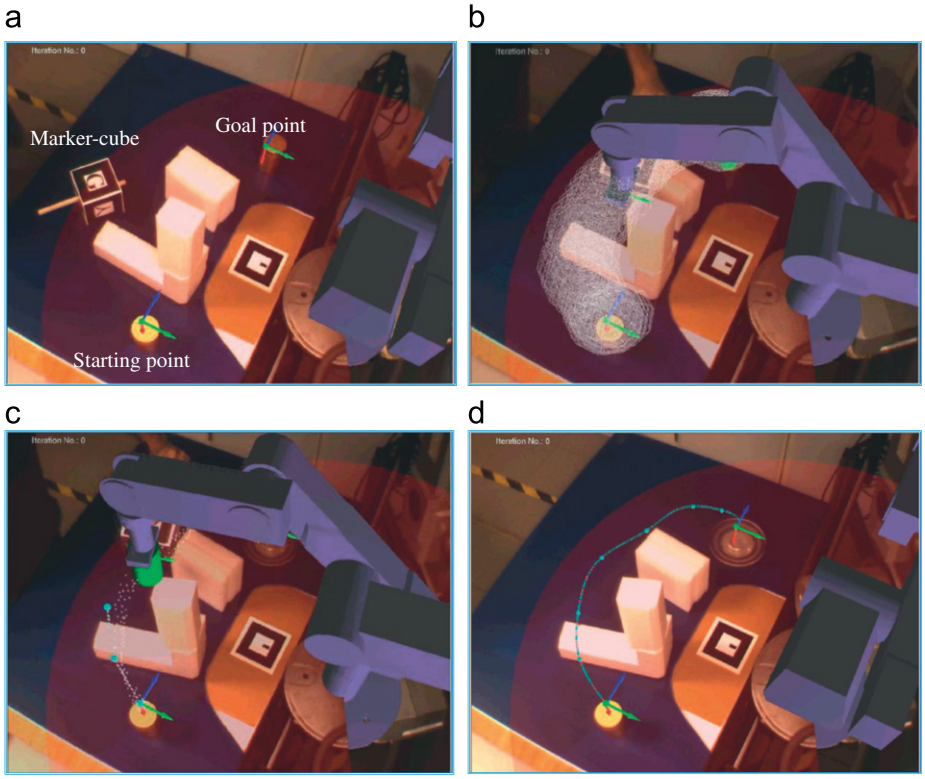

Fig. 8. Geometric path planning in RPAR-II system. (a) An object to be moved from the starting point to the goal point; (b) CFV generation; (c) creation of via-points; and (d) geometric path generated by the starting point, via-points and the goal point.

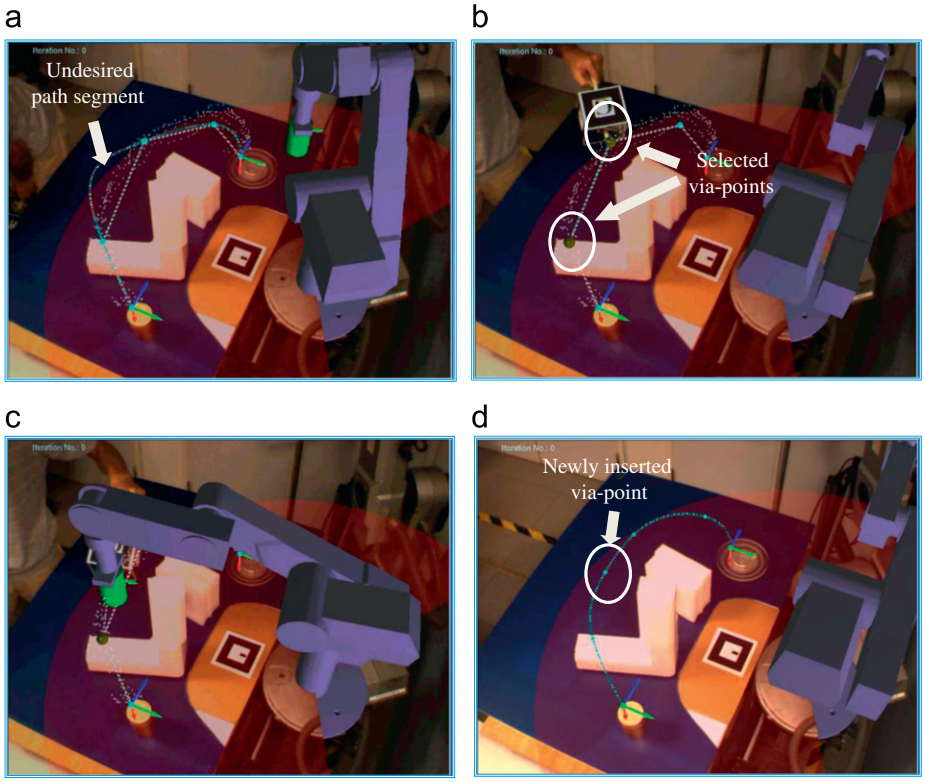

Fig. 9. Modification of via-points. (a) Initial path generated by via-points; (b) adjacent via-points Selection; (c) new via-point insertion; and (d) path re-generation.

- Trajectory simulation
  The interactive simulation enables the users to preview the planned trajectory (Fig. 10(a)) and visualize the deviation of the simulated trajectory from the planned one. At each time instance, the link which is most likely to have deviated from the planned path is highlighted in bright color. Fig. 10(b) and (c) shows that the

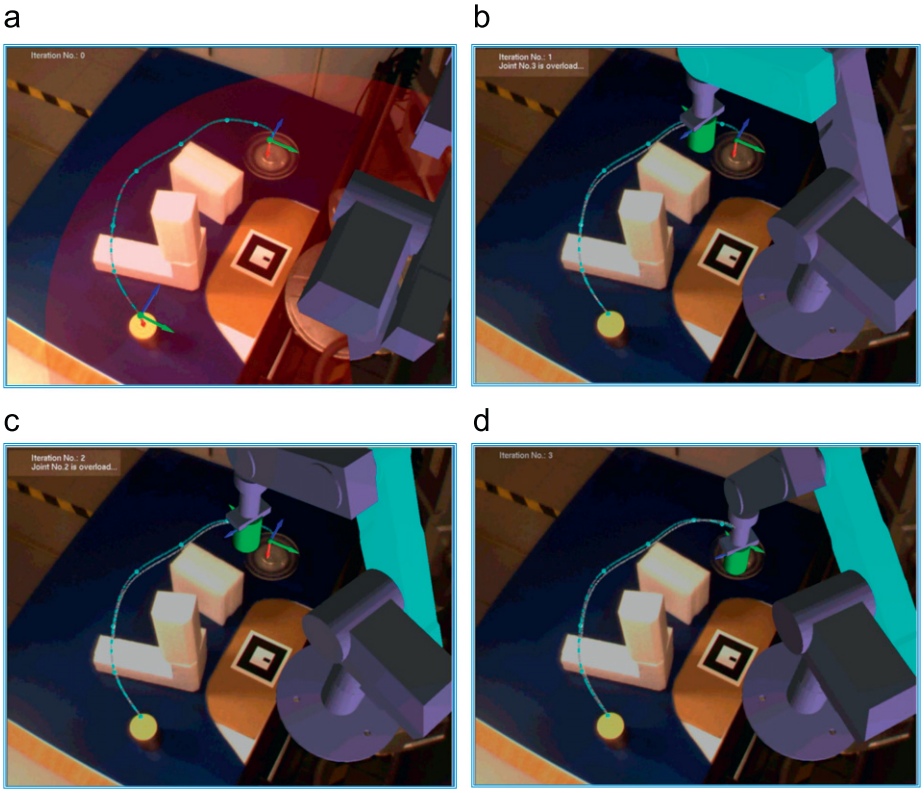

Fig. 10. Trajectory planning and interactive simulation. (a) The planned trajectory registered over the real working environment; (b) interactive trajectory simulation: the first trial (unsuccessful); (c) interactive trajectory simulation: the second trial (unsuccessful); and (d) interactive trajectory simulation: the third trial (successful).

computed torques of joints 3 and 2 have violated their limits; such information is displayed on the screen enabling the users to perceive, and the simulation is halted at the current pose, respectively. Under these two situations, the derivative gains of joint 3 and joint 2 were adjusted gradually. Fig. 10(d) shows a satisfactory simulation after the derivative gains have been tuned twice. Nonetheless, if, after a number of iterations (e.g., 5 iterations), the simulation is still unsatisfactory, the trajectory should be optimized again using a more conservative log-barrier parameter.

6. Conclusions and future work

In this paper, an AR-based system is proposed to assist the users in planning and programming robotic tasks complying with robot dynamics constraints. A virtual robot model, which is a replicate of a real robot, is used to perform and simulate the task planning process. Starting from generating a CFV relevant to the workspace of a given pick-and-place task, through manipulating an interactive device, a user will be able to perform operations such as via-points selection and modification, in order to achieve a smooth and collision-free path.

After generating a cubic-spline path between the starting point and the goal point using the via-points created, a log-barrier approximated optimization scheme is implemented to transfer the geometric path into a time-scale trajectory taking into account the robot actuators constraints. The interactive simulation allows the users to preview, evaluate the optimization outputs, and select a practical trajectory profile taking into consideration both the path duration and actuator usage. Meanwhile, the users can observe the link that is most likely to deviate from the path, and visualize the instantaneous actuator usage when the virtual robot is performing the planned trajectory. In addition, both the control parameters and log-barrier parameter can be tuned to achieve a more desired time-scale trajectory complying with the robot actuator capabilities. The trajectory obtained can then be converted into controller codes and executed on a real robot.

Future work will focus on extending the proposed system for path-following tasks, such as welding, painting, gluing, etc., taking into account additional constraints, such as the velocity and acceleration constraints. In addition, the planned trajectory needs to be transferred into the controller codes and executed on a real robot for further verification. The limitation of the proposed system is the low level of accuracy achievable due to the adoption of the ARToolKit-based tracking scheme. To enhance the accuracy of this system, a more accurate and flexible tracking method will be explored to enhance the HRI and improve the system performance.

References

[1] Pires JN, Veiga G, Araújo R. Programming-by-demonstration in the coworker scenario for SMEs. Industrial Robot: An International Journal 2009;36(1):73–83.
[2] Brogårdh T. Present and future robot control development—an industrial perspective. Annual Reviews in Control 2007;31(1):69–79.
[3] The European Robot Initiative for Strengthening the Competitiveness of SMEs in Manufacturing: The robot capable of understanding human-like instructions. 〈http://www.smerobot.org/15_final_workshop/download/presentations/02_New_devices_and_methods_20090507.pdf〉, last accessed on 27 April 2011.
[4] Chong JWS, Ong SK, Nee AYC, Youcef-Youmi K. Robot programming using augmented reality: an interactive method for planning collision-free paths. Robotics and Computer-Integrated Manufacturing 2009;25(3):689–701.
[5] Zaeh MF, Vogl W. Interactive laser-projection for programming industrial robots. In: Proceedings of the international symposium on mixed and augmented reality. Santa Barbara, CA; 2006. p. 125–8.
[6] Reinhart G, Munzert U, Vogl W. A programming system for robot-based remote-laser-welding with conventional optics. CIRP Annals—Manufacturing Technology 2008;57(1):37–40.
[7] AVILUS. 〈http://www.avilus.de/〉, last accessed on 27 April 2011.

[8] Lieberknecht S, Benhimane S, Meier P, Navab N. A dataset and evaluation methodology for template-based tracking algorithms. In: Proceedings of the international symposium on mixed and augmented reality. Orlando, FL; 2009. p. 145–51.
[9] Hollmann R, Hägele M, Verl A. Learning probabilistic models to enhance the efficiency of programming-by-demonstration for industrial robots. In: Proceedings of the international symposium on robotics. Munich, Germany; 2010.
[10] ARVIKA: 〈http://www.arvika.de/www/pdf/flyer_e.pdf〉, last accessed on 27 April 2011.
[11] Pires JN, Godinho T, Ferreira P. CAD interface for automatic robot welding programming. Industrial Robot: An International Journal 2004;31(1):71–6.
[12] Skourup C, Pretlove J. Intuitive robot programming based on operators' implicit knowledge. In: Proceedings of the international conference on systems, man, and cybernetics, vol. 2. Tucson, AZ; 2001. p. 702–5.
[13] Bicchi A, Peshkin MA, Colgate JE. Safety for Physical Human–Robot Interaction. In: Siciliano B, Khatib O, editors. Springer Handbook of Robotics. Berlin, London: Springer; 2007. p. 1335–48.
[14] Burdea GC. Virtual reality and robotics in medicine. In: Proceedings of the IEEE international workshop on robot and human communication. Tsukuba, Japan; 1996. p. 16–25.
[15] Freund E, Rossmann J. Projective virtual reality: bridging the gap between virtual reality and robotics. IEEE Transactions on Robotics and Automation 1999;15(3):411–22.
[16] Freund E, Rossmann J. Projective virtual reality as a basis for on-line control of complex systems-not only-over the Internet. Journal of Robotic Systems 2005;22(3):147–55.
[17] Liu Z, Bu W, Tan J. Motion navigation for arc welding robots based on feature mapping in a simulation environment. Robotics and Computer-Integrated Manufacturing 2010;26(2):137–44.
[18] Billard A, Calinon S, Dillmann R, Schaal S. Robot programming by demonstration. In: Siciliano B, Khatib O, editors. Springer Handbook of Robotics. Berlin, London: Springer; 2007. p. 1371–94.
[19] Argall BD, Chernova S, Veloso M, Browning B. A Survey of robot learning from demonstration. Robotics and Autonomous Systems 2009;57(5):469–83.
[20] Aleotti J, Caselli S. Robust trajectory learning and approximation for robot programming by demonstration. Robotics and Autonomous Systems 2006;54(5):409–13. (special issue).
[21] Chen J, Zelinsky A. Programming by demonstration: coping with suboptimal teaching actions. The International Journal of Robotics Research 2003;22(5): 299–319.
[22] Atkeson CG, Schaal S. Learning tasks from a single demonstration. In: Proceedings of the IEEE international conference on robotics and automation Albuquerque, vol. 2; 1997. p. 1706–12.
[23] Yanagihara Y, Kakizaki T, Arakawa K, Isoda Y. A multimodal teaching advisor for sensor-enhanced robotic systems in manufacturing. Robotics and Computer-Integrated Manufacturing 1998;14(4):263–73.
[24] Iba S, Paredis CJJ, Khosla PK. Interactive multimodal robot programming. The International Journal of Robotics and Research 2005;24(1):83–104.
[25] Marín R, Sanz PJ, Nebot P, Wirz R. A multimodal interface to control a robot arm via the web: a case study on remote programming. IEEE Transactions on Industrial Electronics 2005;52(6):1506–20.
[26] Schwald B, de Laval B. An augmented reality system for training and assistance to maintenance in the industrial context. Journal of WSCG 2003;11(1): 425–32.
[27] Reinhart G, Patron C. Integrating Augmented Reality in the assembly domain—fundamentals, benefits and applications. CIRP Annals—Manufacturing Technology 2003;52(1):5–8.
[28] Gausemeier J, Fruend J, Matysczok C. AR-planning tool—designing flexible manufacturing systems with Augmented Reality. In: Proceedings of the eurographics workshop on virtual environments. Barcelona, Spain; 2002. p. 19–25.
[29] Rastogi A, Milgram P, Drascic D. Tele-robotic control with stereoscopic augmented reality. SPIE Stereoscopic Displays and Virtual Reality Systems III 1996;2653:115–22.
[30] Chou W, Wang T, Zhang Y. Augmented reality based preoperative planning for robot assisted tele-neurosurgery. In: Proceedings of the IEEE international conference on systems, man and cybernetics. Hague, Netherlands; 2004. p. 2901–6.
[31] Nawab A, Chintamani K, Ellis D, Auner G, Pandya A. Joystick mapped Augmented Reality cues for end-effector controlled tele-operated Robots. In: Proceedings of the IEEE virtual reality conference. Charlotte, NC; 2007. p. 263–6.
[32] Chintamani K, Cao A, Ellis RD, Pandya AK. Improved tele-manipulator navigation during display-control misalignments using Augmented Reality cues. IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans 2010;40(1):29–39.
[33] Bischoff R, Kurth J. Invited talk: concepts, tools and devices for facilitating human–robot interaction with industrial robots through augmented reality. In: Proceedings of the ISMAR workshop on industrial augmented reality. Santa Barbara; 2006.
[34] Craig JJ. Introduction to Robotics, Mechanics and Control. NY: Pearson Education Inc.; 2005.
[35] Bobrow JE, Dubowsky S, Gibson JS. Time-optimal control of robotic manipulators along specified paths. The International Journal of Robotics Research 1985;4(3):3–17.
[36] Shiller Z. On singular time-optimal control along specified paths. IEEE Transactions on Robot Automation 1994;10(4):56–71.
[37] Constantinescu D, Croft EA. Smooth and time-optimal trajectory planning for industrial manipulators along specified paths. Journal of Robotic Systems 2000;17(5):233–49.
[38] Duleba I. Minimum cost, fixed time trajectory planning in robot manipulators. A suboptimal solution. Robotica 1997;15(5):555–62.
[39] Verscheure D, Demeulenaere B, Swevers J, De Schutter J, Diehl M. Time-energy optimal path tracking for robots: a numerically efficient optimization approach. In: Proceedings of the IEEE international workshop on advanced motion control. Trento, Italy; 2008. p. 727–32.
[40] Verscheure D, Diehl M, Schutter De J, Swevers J. On-line time-optimal path tracking for robots. In: Proceedings of the IEEE international conference of robotics and automation. Kobe, Japan; 2009. p. 599–605.
[41] Boyd S, Vandenberghe L. Convex Optimization. Cambridge: University Press; 2004.
[42] Roboop: 〈http://www.cours.polymtl.ca/roboop〉, last accessed on 27 April 2011.
[43] Chan TF, Dubey RV. A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators. IEEE Transactions on Robotics and Automation 1995;11(2):286–92.
[44] gnuplot: 〈http://www.gnuplot.info〉, last accessed on 27 April 2011.