Robotics and Computer-Integrated Manufacturing 25 (2009) 689–701

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

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

Robot programming using augmented reality: An interactive method for planning collision-free paths

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

[a] Innovation in Manufacturing Systems and Technology Programme, Singapore-MIT Alliance, Singapore
[b] Department of Mechanical Engineering, Massachusetts Institute of Technology, USA
[c] Department of Mechanical Engineering, Faculty of Engineering, National University of Singapore, 9 Engineering Drive 1, Singapore 117576, Singapore

ARTICLE INFO

Article history:
Received 17 October 2006
Received in revised form
7 May 2008
Accepted 20 May 2008

Keywords:
Robot programming
Augmented reality
Path planning
Methodology
Interactive
Collision-free volume
Beam search algorithm

ABSTRACT

Current robot programming approaches lack the intuitiveness required for quick and simple applications. As new robotic applications are being identified, there is a greater need to be able to programme robots safely and quickly. This paper discusses the use of an augmented reality (AR) environment for facilitating intuitive robot programming, and presents a novel methodology for planning collision-free paths for an *n*-d.o.f. (degree-of-freedom) manipulator in a 3D AR environment. The methodology is interactive because the human is involved in defining the free space or collision-free volume (CFV), and selecting the start and goal configurations. The methodology uses a heuristic beam search algorithm to generate the paths. A number of possible scenarios are discussed.

1. Introduction

Robots have replaced humans in performing a wide range of tasks which would otherwise be time-consuming and/or dangerous. Industrial robots have long been used to increase productivity and efficiency through the automation of manufacturing processes, such as welding and assembly. Increasingly important in recent years is the possibility of integrating robots into our everyday lives. Examples of existing systems using this class of robots known as the service robots include serial manipulators for pumping fuel into cars, and helping the elderly or disabled persons perform rudimentary daily tasks. These applications have been identified over a decade ago [1]. In a relatively short period of time, the use of service robots in our environments has progressively increased. Hence, there is currently a need to identify new ways of programming robots safely, quickly and more intuitively. These methods should focus on service robots, and at the same time address issues regarding human–robot interaction and traditional programming methods for industrial robots.

There are three types of traditional robot programming methods, namely, lead-through, walk-through and offline programming [2]. These methods have a number of drawbacks. The use of lead-through and walk-through programming poses safety concerns for the users. The walk-through method is only suitable for certain types of robots because the actuator brakes need to be disengaged. Hence, one main 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 programming task for the humans in a safe environment. Natonek et al. [3] and Aleoti et al. [4] described VR systems used for the training of robots for object manipulation in environments that are known *a priori*. However, totally immersive environments (like CAVE™) are costly and complex. Desktop-based offline programming, on the other hand, is less intuitive as compared to the lead-through and walk-through methods. Offline programming is generally restrictive and inflexible because of the need to model the physical entities to match the real environment [5]. In addition, careful calibration and fine-tuning are needed when replicating the simulated environments. Typically, offline programming packages also require the users to master complex programming languages that are software specific and thus, non-generic.

In general, the task of robot programming is time-consuming and unintuitive [6,7]. A simple task that can be performed easily by a human (such as manoeuvring an arm through obstacles or holding a teapot) is often difficult to replicate using a computer-controlled robot. This is because humans possess the ability to

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

0736-5845/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.rcim.2008.05.002

perform complex manipulations without the need to consciously perceive the detailed motion plans [8]. Recently, an approach known as Programming by Demonstration (PbD) has emerged primarily addressing intuitiveness in human–robot interaction. This approach is unique as the robot system *learns*, and *improves* or *duplicates* actions based on human demonstrations. The PbD approach provides the users with an intuitive and fast way of programming by allowing the task to be performed manually and leaving the robot system to observe, follow and learn from sensory information. A PbD system could potentially be as flexible as a human.

The involvement of the human in providing data for PbD applications raises the issues of data acquiring and data processing. The number of times that a user needs to demonstrate a particular task to provide sufficient information for the learning system needs to be determined. Multiple demonstrations are more practical when the planned task is to be executed many times and a higher performance quality is required. On the other hand, a single demonstration is more practical for simple and quick applications [9]. Data obtained through direct human demonstrations may not be optimal and unwanted data have to be removed [10]. Kaiser et al. [11] discussed the sources of sub-optimality that might occur with human demonstrations and the need for a PbD system to provide online adaptation for task refinement. These ideas were then tested for two tasks, inserting a peg into a hole and opening a door [12].

Despite the advancements in PbD, there are still a number of limitations in reported work. First, it is still very difficult to realize PbD for complex contact manipulations, such as grasping assembly actions. However, a useful and successful application of PbD is in path planning where humans are able to navigate a robot through obstacles in unplanned environments [13–15]. Delson and West [13–15] addressed the sub-optimality issue in human demonstrations through identifying a satisfactory robot path that was free of 'wiggles', using a number of repeated human demonstrations and the inherent human variation. A shortest Euclidean distance path can be identified from a range of allowable, obstacle-free demonstrations for a 2D planar space [15]. A path that is not necessarily shortest but depends on a proposed obstacle-free region for a 3D space can be identified as well [14]. Delson and West [11] also differentiated human variations from workpiece variations for constrained motions. However, their work does not provide any other means of evaluating collisions between parts of a robot other than the end-effector, and the obstacles. The approach reported in Refs. [13–15] has further limitations in that multiple demonstrations of a similar nature are needed and the method is limited to a 2D or 3D configuration space or C-space [16,17]. For example, it has been shown that the shortest Euclidean distance path for the end-effector does not necessarily mean the shortest joint travel for an articulated arm [18]. Most PbD literature reports the use of data gloves, tracked grippers, etc., to obtain the data of the end-effector, so as to interpret the user actions [3,4]. However, it should not be assumed that all parts of the robot are always collision-free in all environments.

In the research presented in this paper, an AR environment that facilitates the proposed robot programming methodology and addresses the issues above is presented. The AR environment provides visualization capability that improves the user's interaction with the real world during the path planning process. In the proposed methodology, the user is responsible for providing suitable input data for a path planning algorithm, in an intuitive manner. The algorithm is a heuristic beam search algorithm, which is used to generate paths for an n-d.o.f. robot. Beam search is appealing for its simplicity and has been used to solve problems in job scheduling [19] and assembly line balancing [20]. The beam search algorithm in this paper is similar to automated sampling-based planning algorithms, such as the Expansive-Spaces Trees (EST) [21], which use sampling strategies that focus on the growth of trees towards unexplored areas of the free space that are relevant to reaching the goal. However, the proposed search in this research does not require the construction of a C-space, and uses a different sampling strategy to maintain a number of unique paths. The applications targeted in this research are also simpler and can be easily demonstrated by a human, such as in the case of finding collision-free paths for simple pick-and-place tasks. The primary goal is to make robot path planning and programming as flexible, intuitive and fast as possible. The proposed methodology can be used in static environments (where the obstacles are stationary) that are unknown, and utilizes the versatile capabilities of the humans in scanning, manoeuvring and making sense of a new environment.

The paper is organised as follows: Section 2 discusses the use of an AR environment 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. Section 5 presents and discusses the experimental results. Lastly, Section 6 concludes the paper and proposes future research opportunities.

2. Robot path planning using augmented reality

AR is an emerging technology that is derived from VR. AR is an environment where computer-generated 3D objects are blended (registered) onto a real world scene, to enhance a user's interaction with the real world. An AR system basically involves two major activities, namely: (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 benefits of AR are well documented, and include an increase in task-related intuitiveness, improved user performance, and efficient training of workers [22]. 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 applications such as maintenance [22], manual assembly [23] and computer-assisted surgery [24].

Compared to the applications above, relatively little work has been reported on the application of AR in robot programming. In the domain of telerobotics and robotics, an augmented display can assist the users of the systems [25]. AR offers the capability to visualize the motions and the trajectories of a robot overlaid on to the real environment, and enable the users to intuitively interact with the spatial information [26]. One of the first robotic applications of AR was in telerobotic control [25]. In the work reported by Rastogi and Milgram [27], 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 virtual robot will execute a task for evaluation by the operator, and if it is satisfactory, the task will be transferred to the real robot. AR has also been used in the programming of painting robots [28], 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. The movements of the user are converted into a robot programme. Zaeh and Vogl [26] presented an AR-based approach for intuitive and efficient programming of industrial robots. Through this AR user-interface, tool trajectories and target coordinates can be visualized and manipulated by means of interactive laser projection. Bischoff and Kazi [29] reported an AR-based human–robot

interface for application with real industrial robots. Their work focused on visualizing workflows that could help inexperienced users cope with complex robot operations and programming tasks.

This paper proposes the application of an AR environment that is coupled with a novel methodology for planning collision-free paths for robot programming where the user directly interacts and controls a scalable virtual robot in a real environment. An AR environment is suitable for developing interactive robot programming and complements the notion of PbD. This is because the human is allowed to participate in the path planning process, to perform certain tasks that are difficult to be simulated in a computer, such as recognizing and interpreting an environment. AR also provides the flexibility to augment the view of a user with *visual guidance*. For example, a virtual robot model can be superimposed over the real entities in a real environment, as shown in Fig. 1.

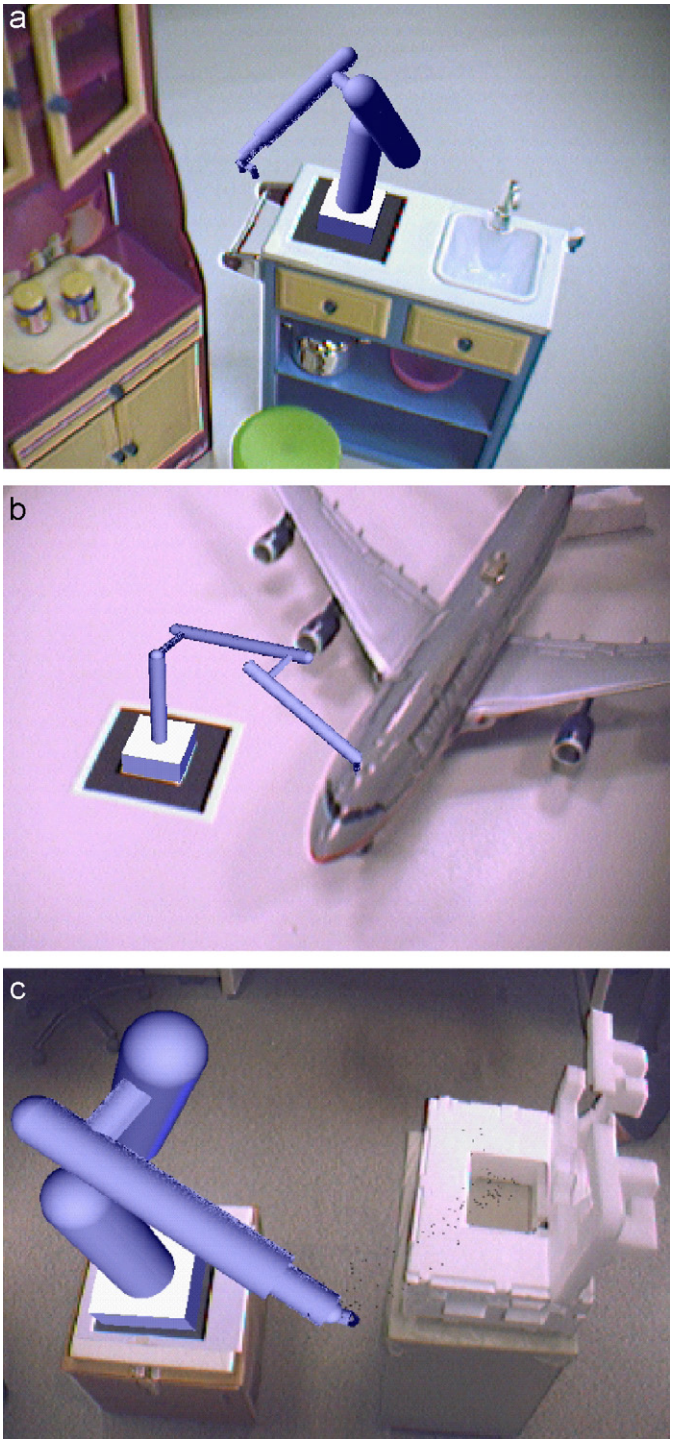

Fig. 1. Virtual robots in real environments: (a) miniature generic-purpose robot in a miniature kitchen, (b) miniature airplane washing robot, and (c) full-scale robot model.

The benefits of the proposed approach, Robot Programming using AR (RPAR), over traditional robot programming approaches are summarized in Table 1. Using a virtual robot means that a programming task can be carried out more safely than traditional online programming methods. An AR environment with scalable robots can also be created in the actual work cell, providing the flexibility and adaptability to different environments when an *in situ* approach is desired. Hence, there is no need to extensively model the environment entities, which is a major disadvantage of VR. An *in situ* approach also does not require careful calibration between the various entities in the environment which is the case when replicating a 'perfectly' modelled environment in VR. Without having to transport a real robot to the actual work cell, RPAR allows the evaluation of various robotic options before the selection is made. The user is able to evaluate simulations of planned paths for different virtual robots. If a robot has been selected, programming can still be performed during the shipment period of the physical robot to the actual work cell. The approach proposed in this research is similar to the intuitive and quick walk-through method as it enables a human to directly move a virtual robot when planning a path. Therefore, the level of intuitiveness is high as compared to lead-through and non-immersive offline programming. Another advantage of the proposed RPAR environment with a virtual robot is the programming of large robots where the walk-through method is infeasible (such as airplane washing robots as shown in Fig. 1b) as the proposed methodology for planning collision-free path and the RPAP approach are scalable.

3. System description

The proposed RPAR system was implemented using the video-based tracking method in ARToolkit [30] and written using the C programming language. The ARToolkit method is based on the identification of 2D square markers with unique patterns printed on them. The HMD of the RPAR system consists of a single IEEE Firefly camera and an i-glasses video display goggle. This setup is based on the inside-out tracking approach where markers attached to objects (static or moving) are tracked using a camera(s) attached on the HMD [24]. The camera(s) not only provides the video images needed for processing, but also provides the user with a view of the real world.

3.1. Relationships between frames

There are three major Euclidean frames in an AR system, namely, the camera, the robot base and the robot wrist, which is a probe, held by the user to teach the robot. As an example in Fig. 2, the **B** marker is used as the base and the **A** marker as the wrist. The relationships between the camera (X_c, Y_c, Z_c), robot base (X_B, Y_B, Z_B) and the wrist (X_W, Y_W, Z_W) frames are obtained using the marker detection method in ARToolkit, while the relationship between the wrist and the end-effector reference point ($X_{W'}$ $Y_{W'}Z_{W'}$) frames is known. The matrix M relates any two frames and consists of the 3×3 rotation, R and 3×1 translation, T matrices. Therefore, the relationship between the robot base, B

Table 1
Comparison between RPAR and traditional robot programming approaches

	Robot type	Safety issue	Environment entities	Evaluation	Intuitiveness
RPAR	Virtual	No	Real	Flexible	High
Lead-through	Real	Yes	Real	Inflexible	Low
Walk-through	Real	Yes	Real	Inflexible	High
Offline (VR)	Virtual	No	Virtual	Flexible	Low

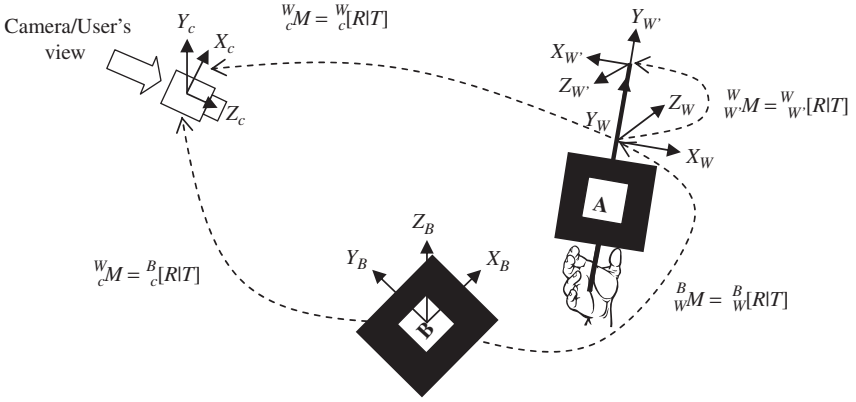

Fig. 2. Relationships between markers and camera.

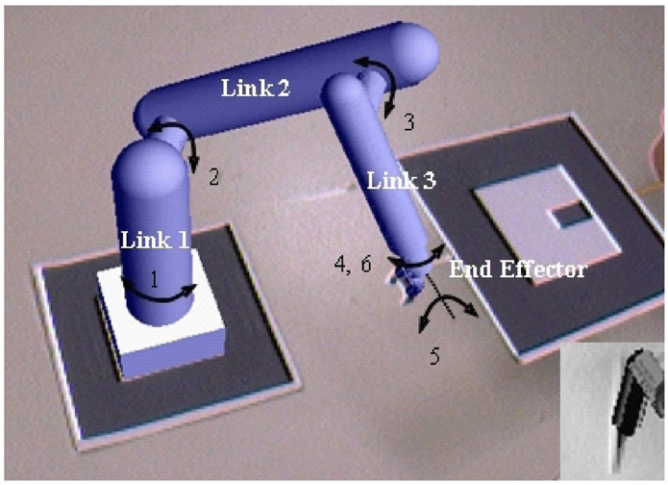

Fig. 3. Six d.o.f. robot model and an alternative end-effector (bottom right).

and the end-effector reference point, W' frames is given by

$$ {}^B_{W'}M = {}^B_cM[{}^W_cM]^{-1}\,{}^W_{W'}M = {}^B_cM^c_W M^W_{W'}M. \quad (1)$$

3.2. Robot kinematics model

The configuration of a robot can be characterized by its generalized coordinates, $\mathbf{q} = [q_1, q_2, \ldots, q_n]$, where n is the number of d.o.f. To illustrate the methodology, a six d.o.f. revolute joint robot (similar to the PUMA 560 industrial robot with three axes intersecting at the wrist [31]) is used, as shown in Fig. 3. The angles of the joints are denoted as $\mathbf{q} = \boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_6]$. Fig. 3 shows an example of a rendered gripper-type end-effector corresponding to the physical probe with the same end-effector type. It is important to note that other than the PUMA 560-type robot in this paper, other robot models can also be used. This provides flexibility for the evaluation of various robotic options before a final decision is made. 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 d.o.f. Avoiding multiple solutions at any one time is essential to ensure the process is intuitive for the human who is moving the robot. If the end-effector is moved out of the workspace of the robot, the robot will stop and remain in the last feasible configuration prior to the workspace violation. Both forward and inverse kinematics solutions used in the RPAR system have been reported by Craig [31].

3.3. System architecture

The basic system architecture is shown in Fig. 4. The input to the inverse kinematics module is the pose of the marker of the wrist, which can be obtained using the pose determination module in ARToolkit. The inverse kinematics module calculates the necessary generalized coordinates of the virtual robot based on this pose, while the forward kinematics module calculates the coordinates of the end-effector reference point based on the generalized coordinates. For each video frame, display of the virtual robot and other virtual elements is provided to the user through the HMD, using the OpenGL Renderer. This virtual display serves as feedback that is useful while visually checking for collisions between the robot and the real physical entities, and inspecting the quality of the planned paths. The torque/force module calculates the amount of work or effort made by the robot for a particular movement, and is used to calculate the path cost (Section 4.3.1). The path is generated by the path planner before being transferred to the physical robot controller (the physical robot in Fig. 4 is a mock-up for illustrative purposes). The physical robot controller is not a part of the current RPAR system.

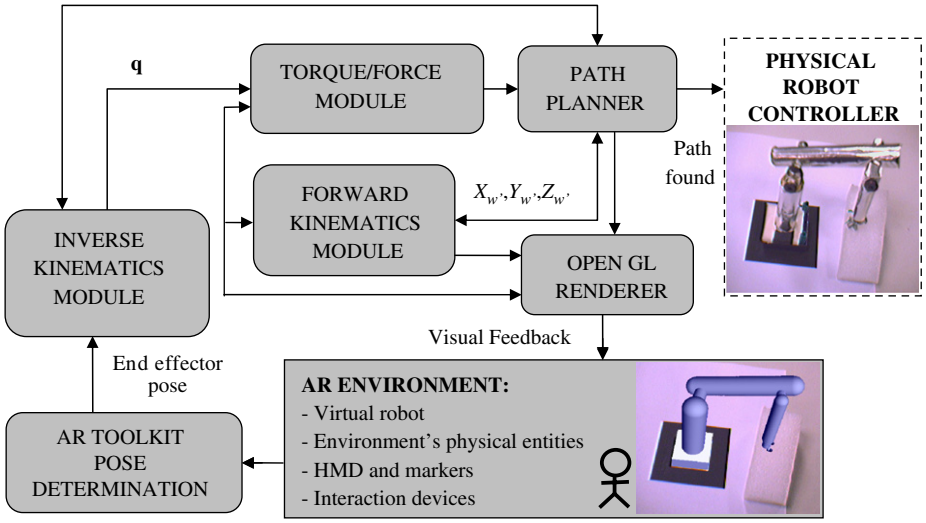

Fig. 4. System architecture (excluding the physical robot controller).

4. The RPAR methodology for path planning

The goal of this research is to generate smooth paths consistently and more reliably as compared to these paths being demonstrated by a human directly. Therefore, the use of a *single* direct demonstration (using the end-effector probe to move the robot along a path) as the final path, is undesirable. This is because the resolution and smoothness of the generalized coordinate profiles for the resulting path will be inconsistent due to either one or a combination of (1) inconsistency in the human hand movements, (2) inconsistency in the sampling rate, and (3) a low sampling rate. Although smoothing can be performed as a post-processing step, it is difficult to determine the appropriate level of smoothness so as to guarantee that the path is collision-free [15].

The proposed approach incorporates the ability of the human to plan collision-free paths quickly. Through moving the end-effector (probe) of a virtual robotic arm, the user is able to move the virtual robot in an environment that consists of real entities to perform the path planning. A continuous collision-free volume (CFV), which defines a subset of the entire free space, is manually generated in a manner similar to an artist 'painting on a 3D canvas'. Therefore, this method is different from the use of multiple direct demonstrations of the same nature to define the free space [13–15]. The CFV allows the search for paths to focus only in areas that are deemed important for the tasks. Unlike offline planners, the CFV avoids the need to model the environment entities because this information is implicit in the volume. Therefore, any part of the robot would avoid obstacles if it remains within the volume.

The proposed methodology is scalable and consists of a number of steps. In the first step, a user has to familiarize himself/herself with the robot and the environment through moving the robot in the environment. In the next step, the user manually generates a CFV using the swept volume of a sphere attached to a probe. The path(s) are planned through interactively selecting valid start and intermediate goal configurations in a sequential manner. A beam search algorithm is used to automatically generate the corresponding collision-free path(s) for the robot joints (as opposed to the single demonstration described earlier). This information is then used to simulate the virtual robot performing the path(s) in the environment, for the user to visually evaluate the smoothness of the path(s) and for any potential collisions. If the results are unsatisfactory, the user may choose to repeat the process. The methodology is shown in Fig. 5.

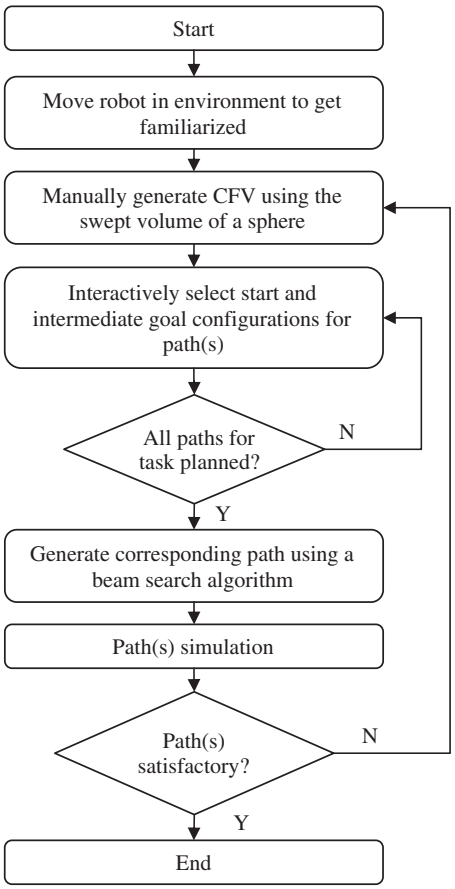

Fig. 5. Proposed RPAR methodology for path planning.

In practice, pick-and-place operations require a robot to manipulate objects such as gripping. In this research, the RPAR methodology is executed only *after* the manipulation of an object has been completed, e.g., an object has been picked up, and this object is to be brought from the current start position to a goal position in the environment. Therefore, the spaces that are occupied by the object at the start configuration and along the path should be part of the CFV. Before the selection of the start

configuration in Fig. 5, the following actions are taken: (1) the object is first moved so that it does not occupy its original space, (2) a CFV is then generated at this previously occupied space, and (3) the object is placed back at its original position after the entire CFV is generated.

4.1. Collision-free volume (CFV) and CFV check

The CFV is generated from the swept volume of a sphere and it is represented as a set of virtual spheres intersecting each other, and displayed to the user through the HMD. Wearing a HMD with the CFV being generated and displayed to him/her in real-time, the user manually generates the CFV in areas which he/she wants the robot to move to through dragging a physical sphere attached to a probe with a marker, as shown in Fig. 6a. There are two radii involved in the generation of the CFV, the virtual sphere radius, r_0 and the physical sphere radius, r. The CFV is a set of p virtual spheres defined by

$$\mathrm{CFV} = \{S_i(c_i, r_0); i = 1, \ldots, p\} \in \Re^3, \quad (2)$$

where S_i is a virtual sphere, characterized by a centre point, c_i, and a radius, r_0.

The 3D coordinates of these centre points are determined with respect to the world coordinate system, which is the coordinate system of the base of the robot (**B** marker in Section 3.1). The choice of the physical sphere's radius, r depends on a number of factors. In practice, the physical sphere used should be larger than the virtual sphere, which is the basic unit of the CFV. Therefore, r is defined by

$$r = r_0 + (\Delta r_{\text{robot\_error}} + \Delta r_{\text{tracking}} + \Delta r_{\text{modelling}}), \quad (3)$$

where $\Delta r_{\text{robot\_error}}$ is the maximum positional error of the physical robot within its workspace, which is available from the manufacturer specifications, $\Delta r_{\text{tracking}}$ is the maximum error of the tracking system, and $\Delta r_{\text{modelling}}$ is the modelling error. The modelling error refers to uncertainties that may exist with the virtual robot model. All the errors are positive.

It should be noted that r depends on the level of resolution required of the environment (whether the sphere is able to access smaller spaces) and a combination of different sphere radii can be used.

Using an AR environment, various options for visual feedback can be employed. The display of the CFV is made semi-transparent to allow the user to view the overall environment at all times, and enable a visual inspection of the quality and compactness of the CFV generated. The virtual robot is rendered during the CFV generation (Fig. 6b) to enable the user to observe whether the robot is able to reach all sections of the CFV, and to check for collisions between the other less critical parts of the robot (for example, links 1 and 2) with any obstacles in the environment.

The quality of the CFV can be evaluated based on its compactness and regularity in shape. The issue of interpolating between samples should not be considered for the CFV generation even though the sampling rate for this video-based system is low. This is because, although interpolation may be useful to obtain a more condensed CFV, it does not guarantee the CFV to be collision-free. There are other ways to obtain a good CFV, such as using a faster sampling rate or having the user move the sphere at a slower velocity. An adequate CFV can be obtained at a low frame rate of 10 fps (frames per second) if the user avoids fast and sudden movements. Lastly, the user can perform multiple passes of the sphere through the CFV to generate a more condensed and regular volume.

After the CFV has been defined, the *CFV Check* is performed to determine whether critical parts of the robot are within this volume and hence collision-free. The choice of critical parts depends on the application at hand. For the examples in this paper, the focus is on the end-effector, as it is essentially the part of a robot most likely to collide with obstacles. Therefore, it requires a more thorough check as compared to other parts of the robot, which can be visually inspected during the simulation. For more complex cases where multiple critical parts need to be considered, a number of separate CFVs can be generated, each handling a particular critical part(s) of the robot. The *CFV Check* is performed through verifying whether a bounding volume for the end-effector is within the CFV. The bounding volume should also consider the dimensions of the object that is being gripped by the end-effector. In this work, a bounding cylinder is used because it is a tighter bound compared to a bounding box and it is also more volume efficient.

4.2. Selection of start and goal configurations

After the CFV has been generated, the user selects start and goal configurations for the path(s) to be planned. This is done by moving the virtual robot using a probe with its tip resembling the end-effector (Fig. 3), and recording the desired configurations. These configurations should be reachable by the robot (which can be verified using the kinematics model as explained earlier), and the critical part or the end-effector should be inside the CFV. To avoid the CFV from blocking the view of the end-effector, the virtual spheres are displayed as smaller dots to indicate where the CFV is so as to guide the user during the selection process. If the end-effector is outside the CFV, the *CFV check* will detect it, and the dots will turn red and a warning will be shown. The proposed approach enables multiple queries to be carried out within the CFV generated, where a query to the system for a path

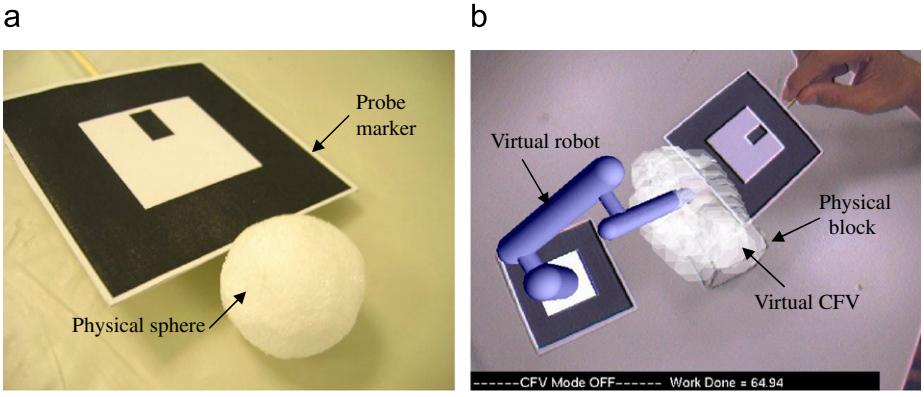

Fig. 6. (a) Sphere attached to a probe with marker and (b) generating a CFV over a rectangular block.

is made every time a pair of start and goal configurations is selected. Therefore, this approach has the characteristics of single query planners such as the EST planner [21] where the focus is on finding a path for a given start and goal configuration pair at any one time, instead of constructing a discrete roadmap of the entire free space [32].

4.3. Search problem

The search problem can be stated as the determination of an optimal or good path in an n-d.o.f. C-space that is characterized by a cost measure (this will be explained in the next section), given the start and goal configurations. The path to be determined and generated should possess the following characteristics (1) smoother and with fewer wiggles than when performed directly by a human, (2) collision-free for critical sections of the robot, i.e., within the CFV, and (3) all configurations along the path are reachable by the robot. The method formulated in this research uses a heuristic to guide the search, so that as few steps as possible are taken to reach the goal (hence providing joint profiles that are as smooth as possible) and a number of best paths are maintained at each search step.

At each current search step, a neighbourhood region for the current node or head node of a path is generated. This region consists of all the possible perturbations on the generalized coordinates $\mathbf{q}$ of the robot. Each perturbation of any particular joint can be expressed as either decreasing, maintaining or increasing the current value by a predefined magnitude $\varepsilon_j > 0$:

$$\Delta q_{j,t} = -\varepsilon_j \text{ or } 0 \text{ or } \varepsilon_j, \quad \in j = 1, \ldots, n, \tag{4}$$

$$\Delta \mathbf{q}_{c,t} = [\Delta q_{1,t}, \Delta q_{2,t}, \ldots, \Delta q_{n,t}], \quad c = 1, \ldots, 3^n, \tag{5}$$

where j represents the robot joints, and c the different perturbation combinations in radians. For n d.o.f., there are 3^n (three possible ways to perturb each joint as shown by Eq. (4)) possible perturbation combinations. In general, the perturbation magnitudes may be different for different joints depending on the resolution of the actuators.

To obtain the neighbourhood region, these perturbations are added to the current nodes of the paths being maintained. For a particular path which consists of one configuration per search step:

$$\mathbf{q}_{t+1} = \mathbf{q}_t + \Delta \mathbf{q}_{c,t}, \tag{6}$$

where $\mathbf{q}_t = [q_{1,t}, q_{2,t}, \ldots, q_{n,t}]$ is the configuration at search step t and $\Delta \mathbf{q}_{c,t}$ is a perturbation combination c chosen by the search at search step t.

The final path returned as the solution consists of the start, intermediate and goal nodes, which are similar to many discrete waypoints in online programming. The start and goal nodes are selected by a user as described in Section 4.2. This path, consisting of m+1 nodes, can be represented by

$$\underbrace{(\text{Start} = \mathbf{q}_{t=0}) \rightarrow (\mathbf{q}_{t=1} = \mathbf{q}_{t=0} + \Delta \mathbf{q}_{c,t=0}) \rightarrow \cdots \rightarrow (\mathbf{q}_{t=m-1} = \mathbf{q}_{t=m-2} + \Delta \mathbf{q}_{c,t=m-2})}_{m \text{ search steps, } m+1 \text{ nodes}} \rightarrow (\mathit{Goal} = \mathbf{q}_{t=m}). \tag{7}$$

4.3.1. Path cost

The path cost will be used by the search to compare the paths during the node filtering process, which is a local optimization step (this will be explained in the next section). It is a measure for evaluating the amount of work or energy needed by the joints to execute a path; this may depend on the robot dynamics model. However, the overall search procedure is independent of the path cost formulation. Therefore, a decoupled trajectory approach is adopted where the geometric path is first planned without considering the suitable velocities and accelerations needed to achieve the desired motions [33]. Robot dynamics may be included later but this is beyond the scope of this paper. To demonstrate the procedure and for the purposes of simplicity, the path cost used in this work is formulated with the assumption that the accelerations and velocities are small and negligible. Hence, the path cost only considers the robot's kinematics and quantifies the effects of gravitational forces (weights of links) on the robot's configurations at each search step.

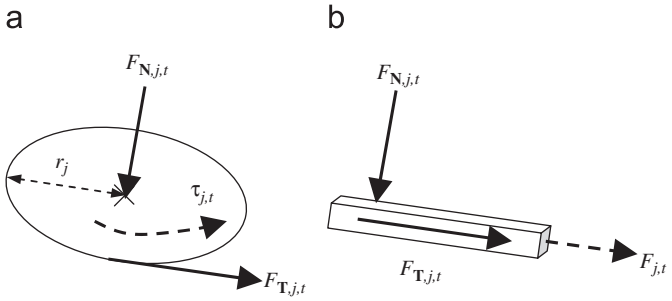

Fig. 7. Forces due to gravitational moments for: (a) rotary joints and (b) linear joints.

For the joint generalized forces, $\mathbf{u} = [u_1, u_2, \ldots, u_n]$ there are two types of joint movements, rotary or linear. The more extended the configuration, the higher the torques or forces at the joints due to higher gravitational moments, compared to a less extended configuration. These gravitational moments are reflected through two force components for each rotary or linear joint, perpendicular (normal) and parallel (tangential) to the movement plane as shown in Fig. 7. Therefore, for each joint j, the generalized force at search step t, is defined as

$$u_{j,t} = \begin{cases} \tau_{j,t} = \mu_j F_{N,j,t} r_j + F_{\mathrm{T},j,t} r_j & \text{if joint } j \text{ is rotational} \\ F_{j,t} = \mu_j F_{N,j,t} + F_{\mathrm{T},j,t} & \text{if joint } j \text{ linear} \end{cases} \quad \in j = 1, \ldots, n, \tag{8}$$

where $\tau_{j,t}$ and $F_{j,t}$ are the torques and forces, respectively, for joint j at search step t; $F_{\mathbf{N},j,t}$ and $F_{\mathbf{T},j,t}$ are the normal and tangential forces, respectively, for joint j at search step t; μ_j is the coefficient of friction at joint j; and r_j is the distance between the rotation centre and the tangential force for rotary joints.

The path cost between two search steps is obtained by multiplying the average torque or force between the two steps, with the magnitudes of the perturbations. For rotary joints, the perturbation is angular, $\Delta q_{j,t} = \Delta \theta_{j,t}$ in radians, and for linear joints, the perturbation is the distance of travel, $\Delta q_{j,t} = \Delta d_{j,t}$ in metres. For a particular path, the *current-path-cost* is defined as the cost from the start, $t = 0$ to a certain search step, $t = T$:

$$\begin{aligned} \mathit{current\text{-}path\text{-}cost} &= \sum_{t=1}^{T} \sum_{j=1}^{n} \frac{u_{j,t} + u_{j,t-1}}{2} |\Delta q_{j,t-1}| \\ &= \sum_{t=1}^{T} \sum_{j=1}^{n} \bar{u}_{j,t} |\Delta q_{j,t-1}|, \end{aligned} \tag{9}$$

where T is the latest or current search step at the current node of the path, $\bar{u}_{j,t}$ is the average generalized force for joint j at search

step t, $\Delta q_{j,t}$ is the perturbation for joint j chosen by the search at search step $t-1$, and *current-path-cost* is in Joules (J).

For Eq. (9), when $T = m$, the *current-path-cost* becomes the *total-path-cost* and both are always positive.

4.3.2. Heuristic beam search

The beam search algorithm uses a heuristic to guide the search towards the goal, and performs local optimization on the path cost along the way. Therefore, the primary concern of the search is getting to the goal configuration followed by improving the path cost during the search whenever possible. In the beam search algorithm, the search space is represented as a tree, and the root node as the start node, S with each node having a branching factor of $b = 3^n$ which is the number of possible perturbation combinations. The size of the search tree is $O(b^h)$, where h is the height of the tree, which is the length of the final path. The beam search approach prunes current nodes that are ranked worse than the best k nodes. The parameter k is the maximum number of nodes that are selected, and is a constant parameter known as the maximum *width of the beam*. The beam search algorithm is attractive because the memory requirements are linear $O(kh)$ rather than exponential as in the case of a full or exhaustive search. The pseudo-codes for the beam search algorithm used in this research are shown in Fig. 8.

The algorithm maintains at most k unique paths during the search for the goal. Line 4 in Fig. 8 perturbs the head nodes or the current nodes of all the paths maintained, to produce $O(3^n k)$ neighbours, which are the 'frontline' of the search. Line 5 filters and deletes the nodes (and their corresponding paths) that are: (1) not within the CFV, (2) configurations that are unreachable by the robot or infeasible, (3) revisited nodes in order to maintain only one path to any particular node, and (4) duplicate nodes where more than one path reach the same neighbour or configuration. The third filter avoids the search from maintaining subset paths of only the greedy path ($k = 1$) and thus stifling the exploration. It also avoids cycling, i.e., perturbation combinations that bring the path to a previous node already existing in the path. When there are duplicate nodes where two or more paths reach the same configuration at the same search step, the fourth filter chooses only the path with the lowest *current-path-cost* among those duplicate paths. This is the local optimization step.

1.Set the first and only path (length = 1) to the start node, *S*.
2. Set saved *goal-path* = NULL.
3. **repeat**
4. Obtain neighbourhood region.
5. Filter current nodes and delete corresponding unwanted paths.
6. Select at most *k* best unique paths using heuristic and delete unwanted paths.
7. **if** any currentnode reaches the goal, *G* and a current *goal-path* is found
8. **if** saved *goal-path* = NULL **or** current *goal-path* cost < saved *goal-path* cost
9. saved *goal-path* = current *goal-path*.
10. Check whether the saved *goal-path* has the lowest cost among all paths.
11. **if** other possibilities of reaching the goal at a lower cost, exist.
12. Delete paths with higher costs than the saved *goal-path*.
13. **else**
14. Return saved *goal-path*.
15. **until** saved *goal-path* is returned.

Fig. 8. Pseudo-codes for the proposed beam search algorithm.

The third filter avoids the search from maintaining subset paths of only the greedy path and thus stifling the exploration. The greedy path is when $k = 1$ and only the single best path is kept. This occurs when nodes are revisited although a previous path to that node already exists as illustrated in the example in Fig. 9. It also avoids cycling, i.e., perturbation combinations that bring the path to a previous node already existing in the path (Fig. 9). When there are duplicate nodes where two or more paths reach the same configuration at the same search step, the fourth filter chooses only the path with the lowest *current-path-cost* among those duplicate paths (Fig. 10). This is the local optimization step.

Line 6 is the essence of the beam search algorithm, where the best k current nodes in terms of the smallest generalized coordinates distance to goal, *dist-to-goal* and their corresponding paths are selected, and the remaining unwanted nodes and their corresponding paths discarded. The *dist-to-goal* for a particular node at search step t is a heuristic defined by

$$\text{dist-to-goal} = \sqrt{\sum_{j=1}^{n}(q_{j,t} - q_{j,G})^2}, \tag{10}$$

where G represents the goal node.

Lines 4–6 in Fig. 8 are repeated until a goal is found, when for any particular current node:

$$|q_{j,t} - q_{j,G}| \leqslant \varepsilon_j, \quad \in j = 1, \ldots, n. \tag{11}$$

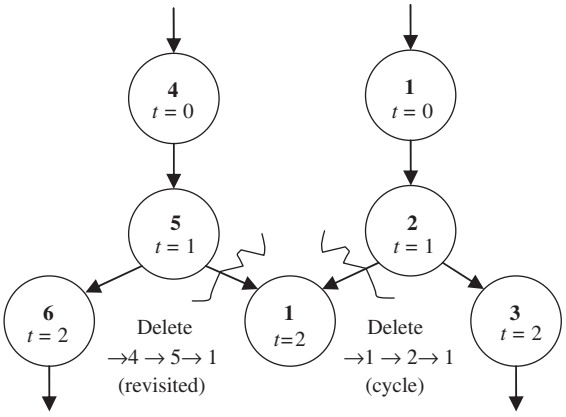

Fig. 9. Filter (3) prunes revisited nodes/states and their corresponding paths (→ 4→5→1), and avoids cycles within a path (→1→2→1).

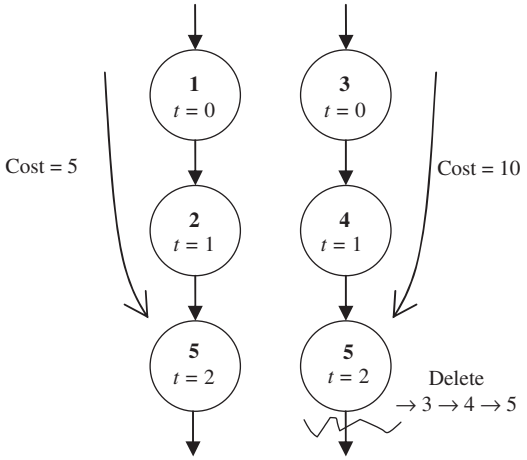

Fig. 10. Filter (4) deletes duplicate nodes (node 5 for path→3→4→5) and selects only the lowest cost path to that particular node.

For Eqs. (10) and (11), if a combination of rotary and linear joints exists, the joint values (and the perturbations) can be scaled so that both rotary and linear joints are within the same order of magnitude.

Lines 7–14 in Fig. 8 verify whether: (1) a current *goal-path* found has the lowest possible cost or (2) there are paths that have not reached the goal, but may have the potential of reaching the goal at a lower cost in the later search process. This is achieved by keeping the paths that have lower *current-path-costs* than the saved *goal-path*, and discarding those paths that have higher *current-path-costs*. When a new or current *goal-path* that has a lower cost than the previous best *goal-path* is found, the saved *goal-path* is updated with this better solution. Eventually, when a current *goal-path* has been found with the lowest cost, and there are no other possible paths that can be found with lower costs than this path in the later search process, this *goal-path* will be returned as the final solution (line 14).

5. Results and discussions

Many experiments have been carried out to observe the effects of generating different CFVs, and varying the beam size, k. To demonstrate the methodology, the 6 d.o.f. rotary robot model ($\mathbf{q} = \boldsymbol{\theta}$ and $n = 6$) in Section 3.2 is used. The calculation of the path cost is based on the actual robot size which is 10 times larger than the scaled down virtual model. The angle perturbations are set to be equal for all joints, i.e., $\varepsilon_j = 0.5^\circ(\pi/180)$ rad, $\in j = 1, \ldots, n$. There are a number of possible scenarios for any particular path planning process.

5.1. Scenario 1: path does not hit the boundary of the CFV

In this scenario, the CFV is relatively compact and regular in shape, such as the example for the scaled down robot shown in Fig. 11. A query is made by recording a pair of valid start and goal configurations. The first and fastest solution path from the defined start configuration to the defined goal configuration is the greedy solution ($k = 1$), and it is optimal in terms of the path cost. The proof is as follows:

Proof I. (Fastest solution is the greedy solution.) Without the loss of generality, take the example of moving from configuration or state $\langle 0,0,0 \rangle$ to $\langle 1,1,1 \rangle$ where the available perturbations at each node are $\langle +1,+1,+1 \rangle$, $\langle +1,0,0 \rangle$, $\langle 0,+1,0 \rangle$, $\langle 0,0,+1 \rangle$, and $\langle 0,0,-1 \rangle$. If the neighbours *are always valid*, then the greedy solution would choose $\langle +1,+1,+1 \rangle$ to reach the goal in *one search step* according to the *dist-to-goal* heuristic in Eq. (10). This is indeed the fastest way of reaching the goal compared to using other neighbours which would take at least three steps (e.g., $\langle 0,+1,0 \rangle \rightarrow \langle +1,0,0 \rangle \rightarrow \langle 0,0,+1 \rangle$ or $\langle 0,+1,0 \rangle \rightarrow \langle +1,0,0 \rangle \rightarrow \langle 0,0,-1 \rangle \rightarrow \langle 0,0,+1 \rangle \rightarrow \langle 0,0,+1 \rangle$). This is confirmed by the final angle profiles for the path (Fig. 12). Since the perturbation magnitudes for all joints are equal, the profiles for each joint are linear with a fixed slope of $\pm\varepsilon_j$/time_step prior to achieving their goal values (slope = 0).

Proof II. (Past cost is a local optimum.) The search only maintains the lowest cost path. If a number of candidate paths reach the same current node, the path that has the lowest *dist-to-goal* at the same search step is kept; this current node is chosen and maintained because $k = 1$. Hence, the path that reaches the goal (current node satisfies Eq. (11)) with the least number of search steps is the smoothest path and a local optimum in terms of the path cost. The path cost in Eq. (9) is a function of the joint travel and is always positive. Hence, smooth paths (fewer search steps) tend to have lower costs as compared to noisy or 'wiggly' paths.

When the path does not hit the boundary of the CFV, the constraints set by the CFV did not influence the solution. Therefore, increasing k does not improve the solution.

5.2. Scenario 2: path hits the boundary of the CFV

This scenario occurs when the boundary of the CFV imposes non-linear constraints that force the joints to re-adjust before achieving their goal values. The boundary is the outer surface of the CFV separating the internal volume from the free space and/or obstacle(s). Hence, the quality of the solution is dependent on the quality of the CFV. When the end-effector hits the boundary of the CFV, there is a possibility of slightly jagged movements in the

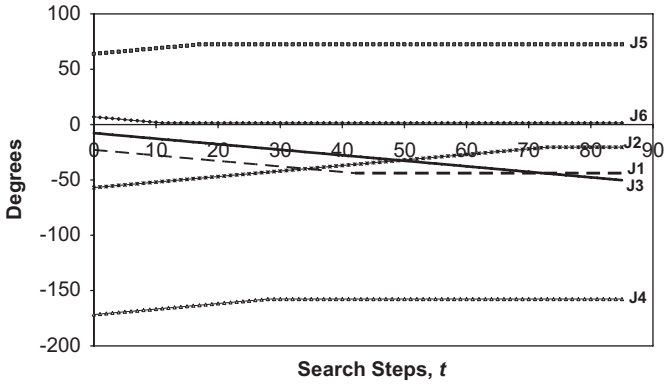

Fig. 12. Solution path for example in Scenario 1 with resolution for all joints = 0.5°.

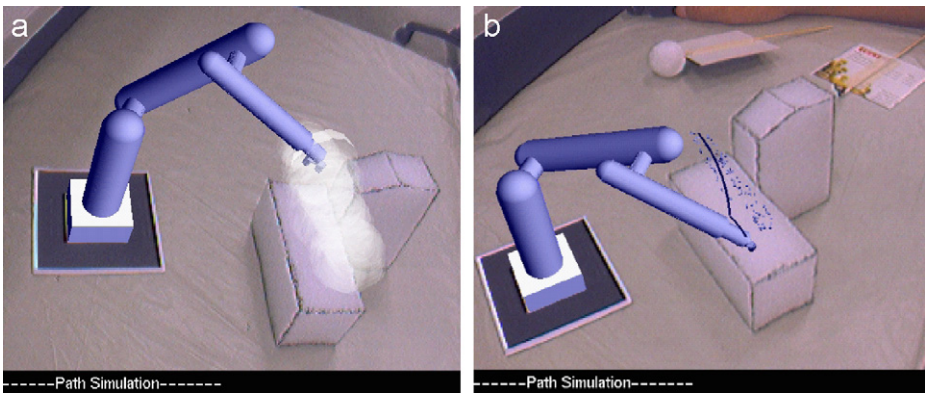

Fig. 11. (a) A well-defined compact CFV and (b) the final path (search steps = 86, *total-path-cost* = 106.0423 J, and final *dist-to-goal* = 0.0075 rad).

angle profiles if the boundary is not smooth as illustrated in Fig. 13. In Fig. 13, when the search results in a path that moves the end-effector from position 1 to position 2, and from position 2 towards the goal G in the direction of the vector 1→2, the end-effector will hit the boundary of the CFV. Hence, the search has to alter its direction upwards and move the end-effector to position 3. However, the quality of the CFV can be improved by using higher sampling rates, moving the physical sphere slowly or using multiple passes of the sphere through the CFV. Using larger magnitudes for the perturbations also avoids the search from being drawn into small areas caused by a non-smooth boundary. However, using lower resolutions will require the assumption that all the intermediate points between the larger steps are collision-free.

A few experiments were carried out to study the effects of increasing k and two examples are shown in Fig. 14. Re-adjustments for certain joints are needed when the end-effector hits the boundary of the CFV. For example, re-adjustments for joints 2 and 3 are needed for Example I, as shown in Fig. 15. Small jagged movements caused by the boundary can also be observed for these joints. However, Fig. 16 shows that increasing k improves the path costs for both Examples I and II. This is because for $k>1$, the search considers paths that are further away from the boundary of the CFV, other than the greedy path which is close to the boundary. These paths are less jagged than the greedy path and have potentially lower costs.

5.3. Scenario 3: the search is stalled

Occasionally, when the CFV contains defects, such as holes, the search may be stalled. These holes are 'artificial obstacles' that may occur when the CFV is not compact enough during the generation process. In such a scenario, most of the search time is spent on trying to find a way around the defect rather than reaching the goal. By increasing k, the availability of several alternative paths improves the search's chances of finding a

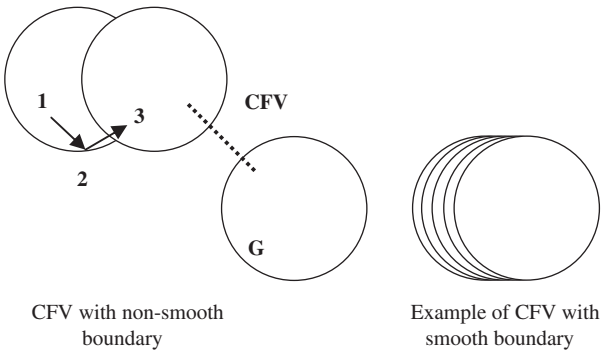

Fig. 13. Jagged movements along the CFV boundary may exist if the CFV is not compact enough, i.e., boundaries are not smooth.

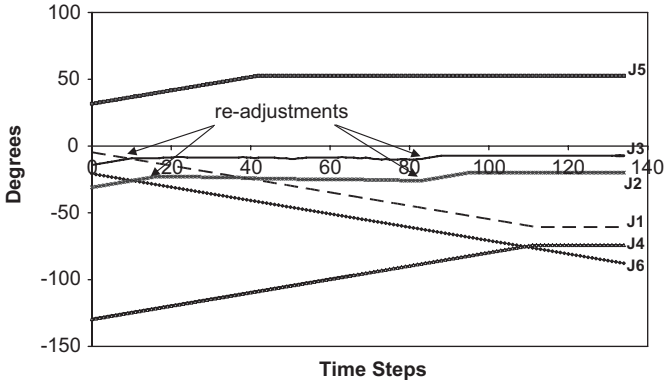

Fig. 15. Solution path for Example I in Scenario 2 with resolution for all joints $=0.5^{\circ}$.

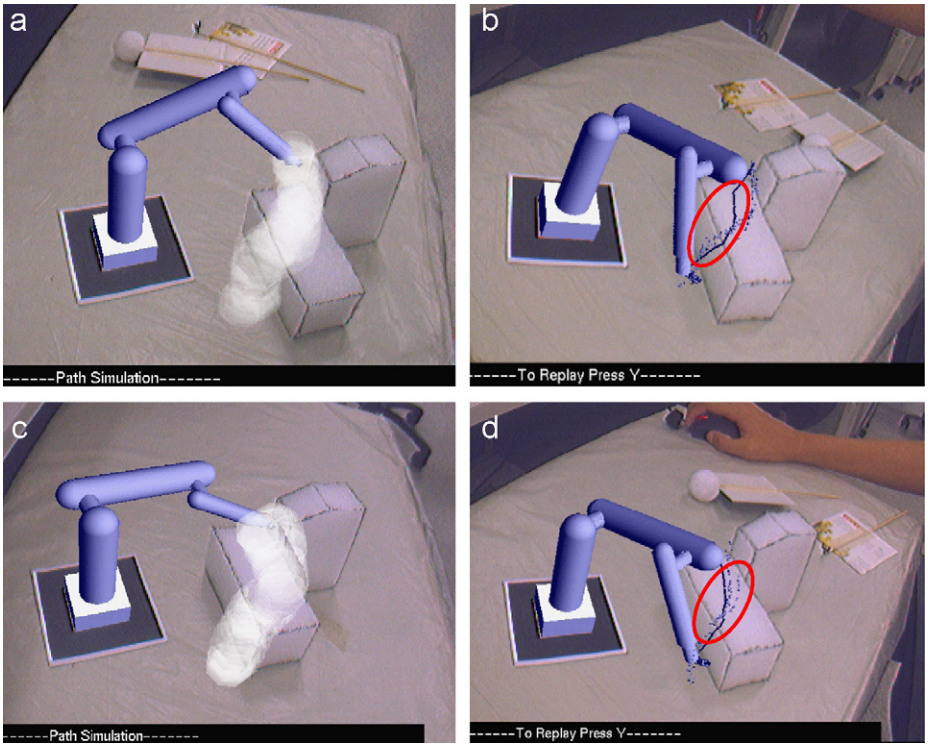

Fig. 14. (a) Example I: when the end-effector hits the boundary of the CFV, (b) the solution path showing the circled portion along the boundary of the CFV, and (c, d) Example II.

solution and therefore, improves its robustness. For the example in Fig. 17, the search was stalled when $k = 1$, 5 and 10 but it was able to avoid the defect entirely when $k \geqslant 20$.

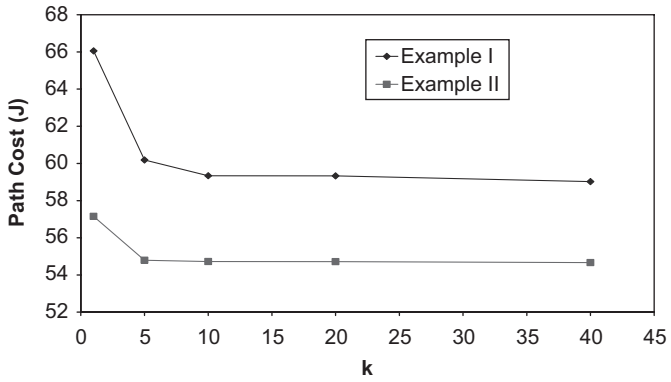

Fig. 16. Effects of increasing k on the path costs for both examples in Scenario 2.

5.4. Other possible scenarios

Besides Scenario 3, the search also fails to find a path for cases where the path is long and relatively winding. However, this can easily be overcome through planning multiple sequential paths. In Fig. 18, the example uses two paths to move the end-effector to a table to pick up a ball and place it on a conveyor. Fig. 19 shows an example of a path planned for a full-scale robot and environment. All these tasks could be planned within 2–3 min.

6. Conclusions

In this paper, a novel approach to robot programming using an AR environment was proposed. The RPAP approach proposed in this paper is scalable, thus offering flexibility and adaptability to different environments when an *in situ* robot programming approach is desired. This RPAP approach allows a user to interactively and directly guide a scalable virtual robot in a real environment to move according to the user's desired operation paths, such that there is no need to extensively model the

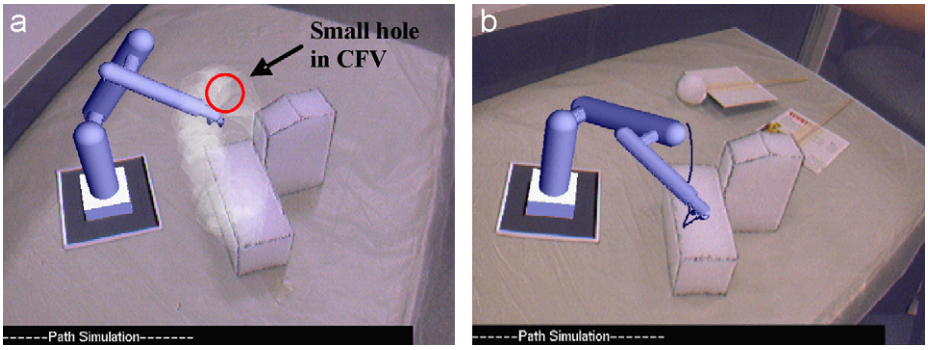

Fig. 17. (a) CFV with defect and (b) path found for $k = 20$ (search steps = 139, *total-path-cost* = 135.1606 J, final *dist-to-goal* = 0.0107 rad, and resolution for all joints = 0.5°).

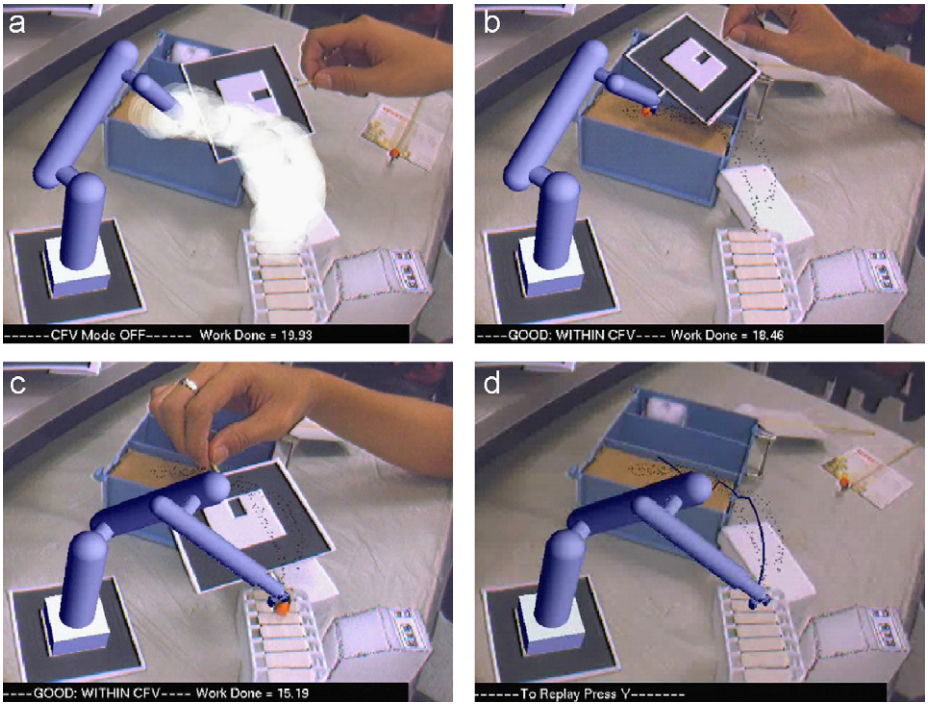

Fig. 18. (a) Generating a CFV, (b) selecting start configuration, (c) selecting final goal configuration, and (d) final paths found.

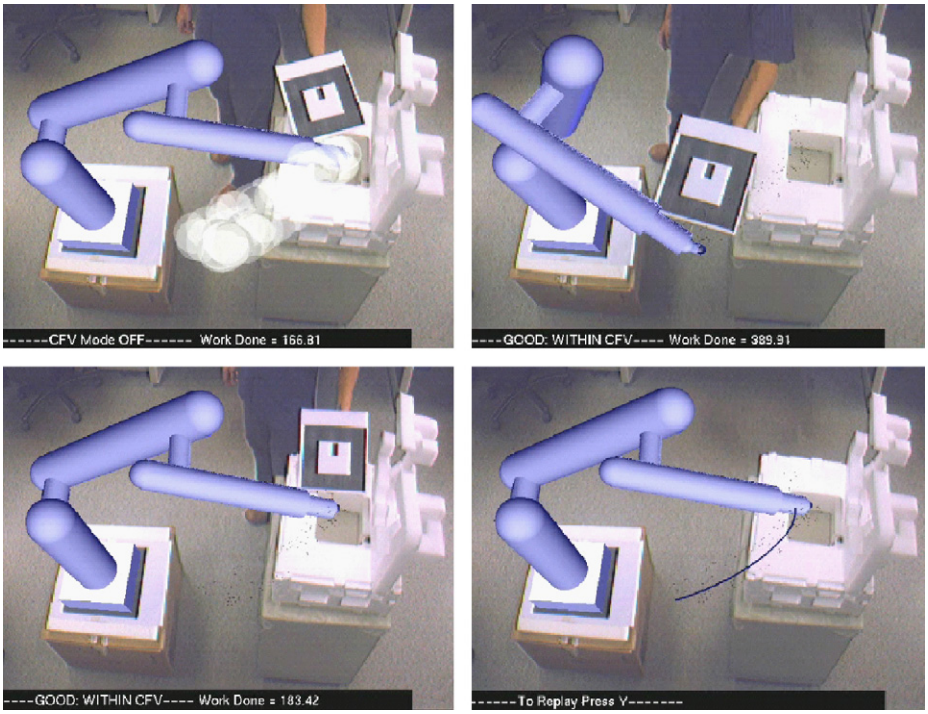

Fig. 19. Planning a path in a full-scale environment.

environment entities. A methodology for planning collision-free paths in the AR environment, in a quick and intuitive way, was also proposed. The path planning methodology included a beam search algorithm to generate paths constrained by an interactively generated CFV. When larger beam sizes were used, the path costs improved and the search was shown to be more robust. From the experiments, it was discovered that the beam size does not need to be too large to achieve these. In the future, alternative methods for generating the CFV and paths can be explored. Currently, new methodologies for robot programming in an AR environment are being developed for applications where the end-effector is required to follow a pre-defined path (e.g., for arc welding). Another issue that needs to be investigated is the level of accuracy achievable using the RPAR methodology. In this work, the ARToolkit tracking method is sufficient to demonstrate the proposed methodology; however, the employment of more sophisticated AR tracking systems with higher accuracy and robustness will greatly increase user acceptance.

References

[1] Schraft RD, Degenhart E, Hägele M, Kahmeyer M. New robot applications in production and service. In: Proceedings of the IEEE international workshop on advanced robotics, Tsukuba, Japan, 8–9 November 1993. p. 15–23.
[2] OSHA technical manual, industrial robots and robot system safety, section IV: chapter 4. Available from: 〈http://www.osha.gov/dts/osta/otm/otm_iv/otm_iv_4.html〉 [last accessed 10.09.06].
[3] Natonek E, Zimmerman T, Fluckiger L. Model based vision as feedback for virtual reality robotics environment. In: Proceedings of the IEEE virtual reality annual international symposium (VRAIS), Los Alamitos, CA, 11–15 March 1995. p. 110–7.
[4] Aleoti J, Caselli S, Reggiani M. Leveraging on a virtual environment for robot programming by demonstration. Rob Autonomous Syst 2004;47(2–3): 153–61.
[5] Robot work cell simulation, University of Missouri, Rolla, MO. Available from: 〈http://web.umr.edu/~vrpl/proj-robot.htm〉 [last accessed 10.09.06].
[6] Pires JF, Godinho T, Ferreira P. CAD interface for automatic robot welding programming. Ind Robot: Int J 2004;31(1):71–6.
[7] Pires JF, Loureiro A, Godinho T, Ferreira P, Fernando B, Morgado F. Object oriented and distributed software applied to industrial robotic welding. Ind Robot: Int J 2002;29(2):149–61.
[8] Asada H, Izumi H. Automatic program generation from teaching data for the hybrid control of robots. IEEE Trans Rob Autom 1989;5(2):166–73.
[9] Atkeson CG, Schaal S. Learning tasks from a single demonstration. In: Proceedings of the IEEE international conference on robotics and automation (ICRA), vol. 2, 20–25 April 1997. p. 1706–12.
[10] Friedrich H, Kaiser M, Dillmann R. What can robots learn from humans? Annu Rev Control 1996;20:167–72.
[11] Kaiser M, Retey A, Dillmann R. Robot skill acquisition via human demonstration. In: Proceedings of the 7th international conference on advanced robotics (ICAR), Sant Feliu de Guixols, Spain, September 1995. p. 763–8.
[12] Kaiser M, Dillmann R. Building elementary robot skills from human demonstration. In: Proceedings of the IEEE international conference on robotics and automation (ICRA), vol. 3, 22–28 April 1996. p. 2700–5.
[13] Delson N, West H. Robot programming by human demonstration: adaptation and inconsistency in constrained motion. In: Proceedings of the IEEE international conference on robotics and automation (ICRA), vol. 1, 22–28 April 1996. p. 30–6.
[14] Delson N, West H. Robot programming by human demonstration: the use of human inconsistency in improving 3D robot trajectories. In: Proceedings of the IEEE/RSJ/GI international conference on intelligent robots and systems (IROS), vol. 2, 12–16 September 1994. p. 1248–55.
[15] Delson N, West H. Robot programming by human demonstration: the use of human variation in identifying obstacle free trajectories. In: Proceedings of the IEEE international conference on robotics and automation (ICRA), vol. 1, 8–13 May 1994. p. 564–71.
[16] Chen J, Zelinsky A. Programming by demonstration: coping with suboptimal teaching actions. Int J Rob Res 2003;22(5):299–319.
[17] Choset H, Lynch KM, Hutchinson S, Kantor G, Burgard W, Kavraki LE, et al. Principles of robot motion: theory, algorithms, and implementations. Cambridge, MA: MIT Press; 2005.
[18] Abdel-Malek LL, Li Z. The application of inverse kinematics in the optimum sequencing of robot tasks. Int J Prod Res 1990;28(1):75–90.
[19] Sabuncuoglu I, Bayiz M. Analysis of reactive scheduling problems in a job shop environment. Eur J Oper Res 2000;126:567–86.
[20] Erel E, Sabuncuoglu I, Sekerci H. Stochastic assembly line balancing using beam search. Int J Prod Res 2005;43(7):1411–26.
[21] Hsu D, Latombe JC, Motwani R. Path planning in expansive configuration spaces. In: Proceedings of the IEEE international conference on robotics and automation (ICRA), vol. 3, 20–25 April 1997. p. 2719–26.
[22] Neumann U, Majoros A. Cognitive, performance, and systems issues for augmented reality applications in manufacturing and maintenance. In: Proceedings of the IEEE virtual reality annual international symposium (VRAIS), Atlanta, GA, 14–18 March 1998. p. 4–11.
[23] Molineros J, Sharma R. Computer vision for guiding manual assembly. In: Proceedings of the 4th IEEE international symposium on assembly and task planning (ISATP), Fukuoka, Japan, 28–29 May 2001. p. 362–8.
[24] Azuma RT. A survey of augmented reality. Presence: Teleoperators Virtual Environ 1997;6(4):355–85.

[25] Milgram P, Zhai S, Drasic D, Grodski JJ. Applications of augmented reality for human–robot communication. In: Proceedings of the 1993 IEEE/RSJ international conference on intelligent robots and systems (IROS 1993), Yokohama, Japan, 20–26 July 1993. p. 1467–72.
[26] Zaeh MF, Vogl W. Interactive laser-projection for programming industrial robots. In: Proceedings of the international symposium on mixed and augmented reality 2006 (ISMAR2006), Santa Barbara, USA, 22–25 October 2006, p. 125–8.
[27] Rastogi A, Milgram P. Augmented telerobotic control: a visual interface for unstructured environments. In: Proceedings of the KBS/robotics conference, Montreal, Canada, 16–18 October 1995. p. 16–18.
[28] Pettersen T, Pretlove J, Skourup C, Engedal T, Lokstad T. Augmented reality for programming industrial robots. In: Proceedings of the international symposium on mixed and augmented reality (ISMAR), Tokyo, Japan, 7–10 October 2003. p. 319–20.
[29] Bischoff R, Kazi A. Perspectives on augmented reality based human–robot interaction with industrial robots. In: Proceedings of the 2004 IEEE/RSJ international conference on intelligent robots and systems (IROS2004), Sendai, Japan, 28 September–2 October 2004. p. 3226–31.
[30] Kato H, Billinghurst M. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In: Proceedings of the 2nd international workshop on augmented reality, San Francisco, CA, October 1999. p. 85–94.
[31] Craig JC. Introduction to robotics, mechanics and control. 2nd ed. Reading, MA: Addison-Wesley; 1989.
[32] Kavraki LE, Svestka P, Latombe JC, Overmaers MH. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Rob Autom 1996;12(4):566–80.
[33] Bobrow JE, Dubowsky S, Gibson JS. Time-optimal control of robotic manipulators along specified paths. Int J Rob Res 1985;4(3):3–17.