USER-SPECIFIED VIRTUAL FIXTURES FOR AUGMENTING HUMAN-ROBOT INTERACTION

By

Aditya Bhowmick

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Computer Science

August 2014

Nashville, Tennessee

Approved:

Nabil Simaan, Ph.D.
Richard Alan Peters, Ph.D.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Professor Nabil Simaan for his support and guidance in finishing this thesis. His perseverance and dedication to research has been a source of inspiration and guideline for me to follow.

I would like to thank all my lab-mates and especially Jason Pile, Long Wang and Haoran Yu for helping me out with debugging the PUMA electronics when it inevitably started giving problems. Without their help, I would have never been able to finish this thesis in time. I would also like to thank my family whose support and encouragement kept me going at all times.

Lastly, I would like to thank the Department of Electrical Engineering and Computer Science and the Department of Mechanical Engineering at Vanderbilt University for the generous financial aid.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

I.1 Virtual Fixtures in Surgical Robotics

Surgical robots have have had a major role in augmenting surgeon capabilities in the past two decades. They allow for safer execution of a surgical path by filtering out the hand tremors of the surgeon and also allow for enhanced kinematic mapping by allowing haptic feedback to the surgeon. One of the areas where surgical robots have had a major impact is minimally invasive surgery(MIS)[3]. MIS is preferable for the patient as it allows for reduced scarring, reduced pain, faster recovery and reduced infection. Traditional MIS tools do not have enough dexterity to perform complex procedures in small confined spaces [4] and result in reduced dexterity and precision, lack of sensory perception and greater cognitive load on the surgeon. This led to the development of robot-assisted surgical systems such as Intuitive Surgical's DaVinci®[5] and Titans SPORT® (Single Port Orifice Robotic Technology) which enhanced the surgeon's dexterity in confined spaces. However, this still places the entire responsibility of carrying out the procedure on the surgeon.

One of the key advantages of Robotic assistance is the accurate execution of surgical plans. Figure I.1 shows a typical surgical workflow with an illustrative example from total hip replacement [6]. The surgical workflow starts with preoperative imaging (Figure I.1-b) followed by surgical planning and then followed by surgical execution (Figure I.1-c). Depending on the type of robot used, the surgical execution may use user input at varying levels of autonomy. As an example, Figure I.1-d shows robotic milling using the Robodoc® compared to manual broaching of the femur.

There are two approaches in reducing the cognitive load and dependence on the surgeon in following a pre-planned surgical path. One approach involves keeping the surgeon as a supervisor and letting the robot carry out the entire procedure autonomously. These surgical robots are termed as *active robots*. One of the earliest examples of an autonomous surgical robot was the Robodoc® used in Hip and Knee Replacement Surgery [6]. The robot and the bone surface are registered precisely to each other and then the surgeon specifies a path based on preoperative CT scans. The robot then mills out the surface intraoperatively with the surgeon present as a supervisor.

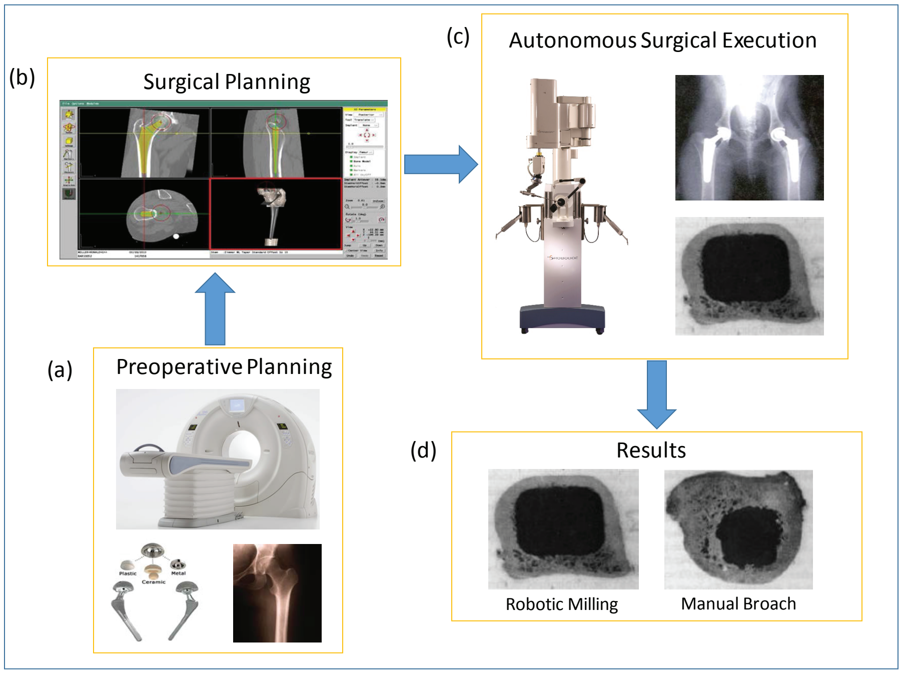


Figure I.1: Robodoc® Surgical System for Hip and Knee Replacement Surgery(Active Surgical System)

The other approach would be to let the surgeon perform the incisions/cuts and let the robot assist the surgeon in guiding the tool. These surgical robots are termed as *semi-active robots*. This offers the surgeon a greater degree of control in the surgical procedure. The assistance provided by the robot to the surgeon is termed as "virtual fixture". One of the earliest implementations of virtual fixtures in surgical robotics was the Acrobot® Surgical System used in Total Knee Replacement(TKR) Surgery [7]. Virtual fixtures were implemented as active constraints that prevented the surgeon from straying into forbidden surgical zones. These constraints were defined by by the surgeon using preoperative CT scans of the patient's leg, Figure I.2. This allowed the surgeon to shape the surface of the knee bones with high precision, which resulted in a highly accurate placement of the knee prosthesis.

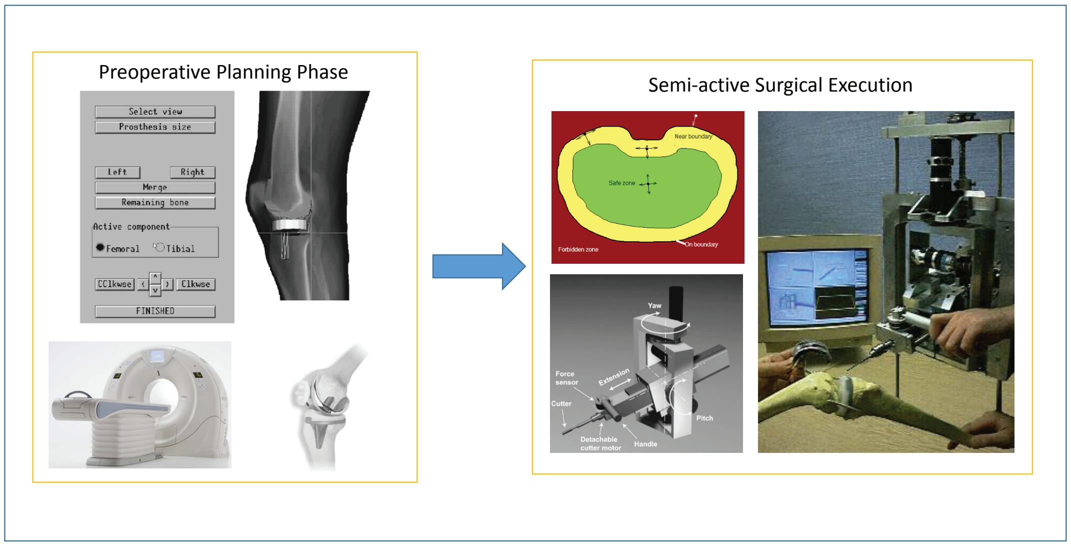


Figure I.2: Acrobot® Surgical System for TKR Surgery(Semi-Active Surgical System)[1, 2]

There has also been research focusing on virtual fixtures in MIS with applications in endoscopic sinus surgery [8] and skull base surgery [9]. The application of virtual fixtures in such constrained areas would be of great assistance to the surgeon in avoiding contact with sensitive anatomy and following a complex surgical path at the same time.

The concept of virtual fixtures can be implemented in many ways. For cooperative control surgical robots (such as the Acrobot® and the JHU Steady Hand Robot), virtual fixtures are implemented using kinematic filtering of the surgeon's hand movement guiding the robot.

In the case of telemanipulated surgical robots (such as the Intuitive Da-Vinci®), there are broadly four telemanipulation controller architectures [10]:

1. *Position Forward (PF)*: where the master is not actuated and the slave just tracks the position of the master
2. *Position Exchange (PE)*: where the slave tracks the position of the master and the master tracks the position of the slave. This scheme of control results in a viscous drag on the master if it tries to lead the slave.
3. *Position Forward/Force Feedback (PFFF)*: where the slave tracks the position of the master, and the forces felt by the slave during its interaction with the environment is fed back to the master
4. *Position Exchange/Force Feedback*: a combination of Position Forward/Force Feedback and Position Exchange.

In this research we broadly classify telemanipulated virtual fixture implementations into three categories:

1. Using virtual fixture force feedback on the master side only
2. Using kinematic filtering of the master commands to the slave robot
3. A combination of both of the above

I.2 Motivation

The challenge with implementing assistive virtual fixtures is the exact coupling of surgical execution to preplanning. This can happen if and only if the anatomy and robot are registered or the preoperative plan is intraoperatively updated to match the anatomy. The task of registering the robot and preoperative data to the area of surgical intervention is a formidable challenge when operating on flexible anatomy. In orthopaedic surgical applications, the bone surface is rigid which allows for accurate registration for defining surgical paths. However, we rarely deal with rigid anatomy when dealing with minimally invasive procedures. For example, in the case of transurethal resection (TUR) of the bladder the anatomy is highly deformable. Registration of the preoperative data with the flexible intraoperative environment is a challenging task which could yield inaccurate results. Most of the previous works have dealt with virtual fixture geometry extracted from preoperative images such as Magnetic Resonant Imaging(MRI) and Computed Tomography(CT) scans [7, 11, 12]. We proposition that most of the previous approaches would not be applicable to highly deformable anatomy such as the bladder. Instead of trying to solve the complex problem of deformable registration, we aim to simplify the problem by proposing an approach which would allow the surgeon to specify motion constraints intraoperatively.

The basic premise is that surgeons are able to visualize the area they want to resect during operation. Therefore, we let the surgeon specify a closed contour depicting the area encompassing the allowable surgical intervention intraoperatively using a visible spectrum laser. These "*user-specified virtual fixtures*" would let the surgeon "draw" the area of interest intraoperatively. We envision that this approach would be more preferable and intuitive to the surgeon.

I.3 Literature Review

Initially the main focus when developing telemanipulated robotic systems for surgery and also in the case of telemanipulated industrial robots, was improving "*telepresence*" [13]. Telepresence as defined by Sheridan [13] is the "visual, kinesthetic, tactile or other sensory feedback from the teleoperator to the human operator such that the human feels that he is present at the remote site, and that the teleoperator is an extension of his own body" [13]. However Rosenberg [14] proposed that sometimes rather than improving the "*fidelity*" of the telepresence, corrupting it is also beneficial. The concept of virtual fixtures was thus defined as the "abstract sensory information overlaid on top of reflected sensory feedback from a remote environment". Rosenberg stated that implementing virtual fixtures improved operator performance by up to 70% [14].

Virtual fixtures can broadly be classified as "barrier virtual fixtures" or "guidance virtual fixtures". The objective with barrier virtual fixtures is to prevent the erroneous tool excursions by the user. In the case of guidance virtual fixtures, the objective is to assist the surgeon in following a curve or a plane and reorient the tool so that the surgeon is able to satisfy both anatomical and geometric constraints.

The development of Acrobot used in TKR surgery was the first implementation of active constraints(barrier virtual fixtures) on a cooperative manipulation robot [7]. The work focused on the advantages of using a semi-active and cooperatively controlled robot in surgery which gave better control over the surgical procedure and enabled the surgeon to shape the bones with greater accuracy and precision . The basic idea behind implementing virtual fixtures as active constraints was to gradually increase the stiffness of the robot as it reached the workspace boundary. During cadaveric studies, registration between the bone surface the preoperative CT scans was done using fiducial markers. Using fiducial markers was unacceptable during clinical trials and therefore the authors came up with a registration method based on the Iterative Closest Point (ICP) algorithm. The bone surface was held fixed in place with respect to the robot by using clamps which was used to register the robot to the anatomy. The surgeon then selected multiple landmark points on the anatomy which was used to generate the initial estimate of the registration between the preoperative images and the bone surface. After that multiple points were selected on the bone surface and registered to the CT scan which was verified by the surgeon. This registration approach would not translate well to other surgical procedures as we would not always have a rigid bone surface easily accessible to the surgeon.

One of the earliest works involving the use of virtual fixtures in MIS was in cardiac surgery [15]. Preoperative CT scans were taken to determine the location of the internal mammary artery (IMA). During the surgery the patient's anatomy was registered to the imaging data and then virtual fixtures was implemented to constrain the motions along adjacent paths to the artery. Hein et. al. [16] also implemented virtual fixtures in the form of workspace restrictions in shaping of the spinal vertebrae.

Abbott et.al. [10] analysed various virtual fixture architectures for telemanipulation. The goal was to determine the best combination of master and slave forbidden region virtual fixtures (FRVF) on different telemanipulator controllers. The conclusion was that the performance is more similar across all telemanipulator architectures however different requirements ("tracking, safety and submittance") called for different implementations of FRVF. The only definitive conclusion was that implementing a strong FRVF at the slave side coupled with no FRVF at the master led to poor telepresence.

Bettini et.al. [11] implemented a system where vision was used to follow a reference trajectory. The type of virtual fixture implemented was termed as "guidance" virtual fixtures as the user was guided to follow a reference trajectory or guided to a reference point. Vision was used to determine the location of the robot with respect to its environment. The effect of varying compliances to get "soft" virtual fixtures was also studied. It was found that soft virtual fixtures provided the user with sufficient guidance to follow a path or move to a point with accuracy and yet have enough control to pull away from the guided path to avoid obstacles.

Kragic et.al. [17] implemented guidance virtual fixtures with the JHU Steady Hand Robot which is a cooperative manipulator i.e. an admittance controlled robot. They also implemented an on-line task recognition system based on Hidden Markov Models(HMM). This approach resembled the work of Rosen [18, 19] on intent identification using HMM's. The system developed was able to recognize if the user intended to avoid the curve and switch off virtual fixtures automatically. The recognition algorithm was trained on different sine curves and was found to be robust with an average accuracy greater than 90%. The implementation of virtual fixtures with HMM allowed the user to avoid a curve with greater ease and was found to lower the total task execution time.

Most of the previous works dealt with implementing virtual fixtures on admittance control robots. Abbott et.al. [20] implemented virtual fixtures on impedance type devices using admittance control. The system implemented did not utilize a force sensor to generate the reference velocities. The methodology was to read in the position error from a predefined set-point and then approximate the position error to a force reading. This force reading was then used to define a reference velocity which was integrated to update the set position. The importance of this work lies in its application to existing impedance type teleoperation systems such as Intuitive Surgical's DaVinci [5].

Marayong et.al. [21] discussed the concept of using virtual fixtures to implement spatial motion constraints. This paper gave a rigorous theoretical definition of virtual fixtures as geometric constraints. Basically, a basis of preferred directions are created off-line to constrain the user along a path. Additionally, the concept of closed loop virtual fixture was also explored which in addition to guiding the user along preferred directions also gets the user back to the constrained curve. It was experimentally verified that closed loop virtual fixtures did not deviate from the required path in both rotational and translational virtual fixtures.

A clinical application for ENT surgery based on spatial motion constraints was developed in [22, 12, 8]. In ENT surgery, especially sinus surgery the operating space for the surgeon is limited. The challenge in these surgeries is to be able to follow a preplanned surgical path in a confined area ("tool tip motion constraints") while ensuring that the tool shaft boundary does not come in contact with the nasal and sinus bones ("anatomic constraints"). The anatomic constraints are generated by the 3D model of the anatomy and the tool tip trajectory is pre-specified. The approach is to map the tool tip motion and boundary information to joint displacements. A constrained optimization algorithm is then used to determine the optimal joint displacements which satisfy the tip motion and anatomic constraints. The 3D model of the skull was generated using a software called 3D-slicer. The skull model was registered to the CT images and the robot using fiducials embedded in the skull. These fiducials were sampled using an Optotrak pointer(which is a 3D point tracking system). The target path was specified with respect to the CT images by tracing a wire embedded in the skull phantom using the Optotrak system. The sampled points were then interpolated using B-splines to get the 3D curve. They also compared the performances between "free-hand mode", "SHR guided hands-on cooperative mode" and "SHR guided remote teleoperation mode"[8]. The results indicated that both the robot guided modes were better than the free-hand mode. The hands-on cooperative mode resulted in a slightly reduced error compared to the teleoperated mode. However, there was no significant difference between the two modes as indicated by the paired t-test. The execution time in the case of the hands-on cooperative mode was however significantly less than the teleoperated mode.

The contribution of this thesis would be in the implementation of a library of virtual fixture primitives that could be expanded to include more complex definitions. We also propose the concept of user specified virtual fixtures which would allow the surgeon to specify the allowable area of surgical intervention intraoperatively using a visible spectrum laser.

I.4 Outline of this work

The primary contributions of this work would be in laying out the framework and foundations behind assistive manipulation algorithms suitable for both cooperative manipulation and telemanipulation controller architectures. The task would be to define a library of virtual fixture primitives, which would allow definitions of new virtual fixtures. This work would also analyse the operator performance improvements by using virtual fixtures. A framework and setup for specifying user-specified virtual fixtures would also be proposed which would also take care of visual registration of the area of surgical intervention to the robot.

First, chapter II presents the theoretical background behind projections which are used as a tool for virtual fixture specification which would be used to derive control equations for virtual fixtures in three cases. Chapter III details the experimental setup used in the experiments conducted. Chapter IV explains the user-control interface which integrates various control modes of the PUMA robot and also detail the safe transition logic between the different operating modes. Finally, chapter V illustrates our experimental results and validates the approach used.

CHAPTER II

THEORETICAL BACKGROUND

II.1 Human-Robot Interaction Modes

Before we move on to the mathematical background behind virtual fixtures, we should first discuss the different manipulation architectures. Essentially, there are two modes of user-controlled manipulation that are used in robotics; *telemanipulation* and *cooperative manipulation*.

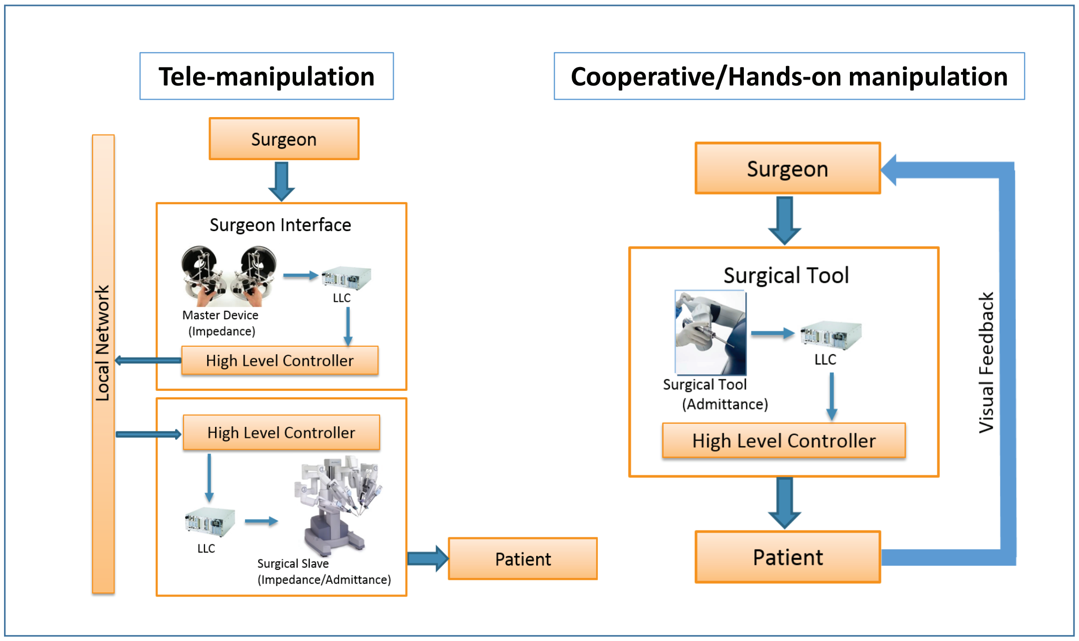


Figure II.1: Interaction modes for manipulation

However, before we define these terms we should talk about *impedance* and *admittance* type devices. Basically, impedance type devices are back-drivable with current (torque) low-level control and accurate dynamics whereas admittance type devices are non back-drivable with voltage (speed) low-level control where the dynamics are usually attenuated by high gear ratios.

Generally we tend to use impedance masters in the case of telemanipulation. Admittance masters are generally encountered in cooperative/hands-on manipulation where the user directly manipulates the robot by applying force to the tool. Impedance slaves are used in surgical systems such as the Da-Vinci. The advantage with using impedance slaves over admittance slaves is the ability to sense forces through motor currents(torque) or joint errors whereas admittance slaves require the use of a force sensor in order to sense forces.

A "*telemanipulation*" system generally consists of an impedance master device, impedance/admittance slave device and a communication network as shown in the figure. The surgeon controls the master device which generates electrical signals from the low level controller. This data is then fed into a high level controller which computes the position and orientation of the master device and sends it to the slave device where this data is processed by its own high level controller. The high-level controller on the slave side then sends the required signal to the low-level controller which moves the slave robot. Intuitive Surgical's DaVinci® is an example of a telemanipulated surgical system.

As mentioned before there are broadly four telemanipulation controller architectures; *Position Forward(PF)* where the slave tracks the master, *Position Exchange(PE)* where the slave tracks the master and the master tracks the slave, *Position Forward/Force Feedback(PFFF)* where the slave tracks the position of the master and the forces felt by the slave are fed back to the master, *Position Exchange/Force Feedback* which is a combination of PFFF and PE.

In contrast, in the case of a "*cooperative/hands-on manipulation*" system the user applies force directly to a tool attached to an admittance master. The forces are sent to the high-level controller which generates the low-level signals which is fed to the low-level controller which moves the tool in the direction of applied force. The Acrobot® Surgical System is an example of a cooperative/hands-on manipulation system.

II.2 Inverse Kinematics Resolved rates control of serial robots

For serial robots, the inverse kinematics problem (finding the joint values given the end effector position and orientation) is nonlinear and difficult to solve in closed-form. We solve this problem in real-time with the use of the "resolved rates" algorithm. Basically, if the robot has a current pose $\mathbf{x_c}$ and we have a desired pose $\mathbf{x_d}$, then the resolved rates algorithm would generate a sequence of joint values $\mathbf{q}$ that would let us reach the desired pose $\mathbf{x_d}$ from the current pose $\mathbf{x_c}$ in a smooth motion.

Assume that the robot has a home configuration, with known joint values $\mathbf{q}_h$. We can easily compute the position and orientation of the end effector $\mathbf{x_h}$ from the direct kinematics. At each time step t, the robot has a current pose $\mathbf{x_c}$ which is different from the desired pose $\mathbf{x_d}$. The difference between the desired pose and the current pose defines the position and orientation error. As the pose $\mathbf{x}$ is defined using the Cartesian position $\mathbf{p} = [p_x, p_y, p_z]$ and a vector of Euler angles $\boldsymbol{\xi} = [\varphi, \theta, \phi] \in \mathbb{R}^3$, it is better to compute the position and orientation errors separately.

The position error δ_p is computed as,

$$\delta_p = \sqrt{(\mathbf{p_d} - \mathbf{p_c})^T(\mathbf{p_d} - \mathbf{p_c})} \tag{II.1}$$

To compute the orientation error, we first need to compute the Rotation $\mathbf{R_e}$ that would fix the orientation error. We have the desired orientation of the end effector in world frame as $\mathbf{R_d}$ and the current orientation of the end effector as $\mathbf{R_c}$. The rotation $\mathbf{R_e}$ to bring $\mathbf{R_c}$ to $\mathbf{R_d}$ is computed as (assuming a fixed frame rotation sequence),

$$\begin{aligned} \mathbf{R_d} &= \mathbf{R_e}(\theta_e, \hat{\mathbf{m}}_e) * \mathbf{R_c} \\ \mathbf{R_e} &= \mathbf{R_d} * \mathbf{R_c}^T \end{aligned} \tag{II.2}$$

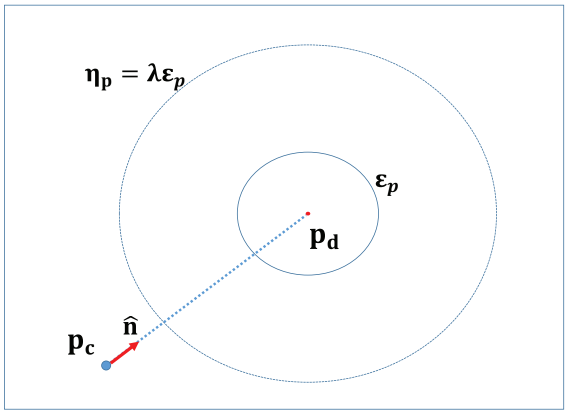
Once we have the rotation $\mathbf{R_e}$, we can compute the axis $\hat{\mathbf{m}}_e$ and angle θ_e for the axis-angle notation using the following equations,

$$\begin{aligned} \theta_e &= \cos^{-1}\frac{trace(\mathbf{R_e}) - 1}{2} \\ \hat{\mathbf{m}}_e &= \frac{1}{2\sin(\theta_e)}\begin{bmatrix} \mathbf{R}_e(3,2) - \mathbf{R}_e(2,3) \\ \mathbf{R}_e(1,3) - \mathbf{R}_e(3,1) \\ \mathbf{R}_e(2,1) - \mathbf{R}_e(1,2) \end{bmatrix} \end{aligned} \tag{II.3}$$

Using the axis-angle representation, we can compute the orientation error δ_ξ as follows,

$$\delta_\xi = \sqrt{(\xi_\mathbf{d} - \xi_\mathbf{c})^T(\xi_\mathbf{d} - \xi_\mathbf{c})} \tag{II.4}$$

Using the position δ_p and orientation error δ_ξ, we can compute the desired twist (linear and angular velocity) $\dot{\mathbf{x}}_d = [\dot{\mathbf{p}}_d, \dot{\boldsymbol{\xi}}_d]$. If we use this twist to compute the joint velocities $\dot{\mathbf{q}}$, we would move towards the desired pose $\mathbf{x_d}$.



$\boldsymbol{\eta}_\mathbf{p} = \boldsymbol{\lambda\varepsilon}_p$
$\boldsymbol{\varepsilon}_p$
$\mathbf{p_d}$
$\hat{\mathbf{n}}$
$\mathbf{p_c}$

Figure II.2: An example of resolved rates algorithm for following a straight line

To ensure that the robot smoothly converges to the desired pose $\mathbf{x_d}$, we make use of a scaling term λ which is used to compute the radius of position error $\eta_p = \lambda\varepsilon_p$ and the radius of orientation error $\eta_\xi = \lambda\varepsilon_\xi$ beyond which the robot moves at the maximum linear and angular speed respectively. In Figure II.2, the robot would move with a maximum speed towards the desired point and then start slowing down when it is within the radius η_p. The desired linear speed $\dot{\mathbf{p}}_d$ and desired vector of euler angles' rates $\dot{\boldsymbol{\xi}}_d$ can be computed as follows,

$$\dot{\mathbf{p}}_d = \|\tilde{v}\|\hat{\mathbf{n}} \tag{II.5}$$

$$\text{where,}\quad \hat{\mathbf{n}} = \frac{\mathbf{p}_d - \mathbf{p}_c}{\|\mathbf{p}_d - \mathbf{p}_d\|},$$

$$\|\tilde{v}\| = \begin{cases} v_{\max}, & \text{if } \frac{\delta_p}{\varepsilon_p} > \lambda, \quad \lambda > 1 \\ \frac{v_{\max}-v_{\min}}{\varepsilon_p(\lambda-1)}(\delta_p - \varepsilon_p) + v_{\min}, & \text{if } \frac{\delta_p}{\varepsilon_p} \le \lambda, \quad \lambda > 1 \end{cases}$$

$$\dot{\boldsymbol{\xi}}_d = \|\dot{\xi}\|\hat{\mathbf{m}}_e \tag{II.6}$$

$$\|\dot{\xi}\| = \begin{cases} \dot{\xi}_{\max}, & \text{if } \frac{\delta_\xi}{\varepsilon_\xi} > \lambda, \quad \lambda > 1 \\ \frac{\dot{\xi}_{\max}-\dot{\xi}_{\min}}{\varepsilon_\xi(\lambda-1)}(\delta_\xi - \varepsilon_\xi) + \dot{\xi}_{\min}, & \text{if } \frac{\delta_\xi}{\varepsilon_\xi} \leq \lambda, \quad \lambda > 1 \end{cases}$$

where $\hat{\mathbf{m}}_e$ is given by Eq. II.3. To compute the joint velocities $\dot{\mathbf{q}}_d$ from the desired twist $\dot{\mathbf{x}}_d$ we make use of the generalized pseudo-inverse $\mathbf{J}^\dagger$ of the Jacobian matrix. The generalized pseudo-inverse is exactly the inverse of $\mathbf{J}$ if the number of joints m is equal to the degree of freedom n. Otherwise, in the neighbourhood of singularity, $\mathbf{J}^\dagger$ is computed based on the singularity-robust inverse as,

$$\dot{\mathbf{q}}_d = \mathbf{J}^\dagger\dot{\mathbf{x}}_d \tag{II.7}$$

where $\mathbf{J}^\dagger = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T + \rho\mathbf{I})^{-1}$ and ρ is a small number and $\mathbf{I}$ is the identity matrix.

Once the desired joint speeds $\dot{\mathbf{q}}_d$ are computed, we compute the new joint values for the next iteration by assuming that the desired joint speed was applied for an increment of time Δt using the following equation.

$$\mathbf{q}_i = \mathbf{q}_{i-1} + \dot{\mathbf{q}}_d\Delta t \tag{II.8}$$

Using the updated joint values, the position δ_p and orientation errors δ_ξ are computed again, using which the desired twist (linear and angular velocity) $\dot{\mathbf{x}}_d = [\dot{\mathbf{p}}_d, \dot{\boldsymbol{\xi}}_d]$ is computed and finally the joint values at the next time step is computed. We continue these steps until we converge to the desired position $\mathbf{x}_d$.

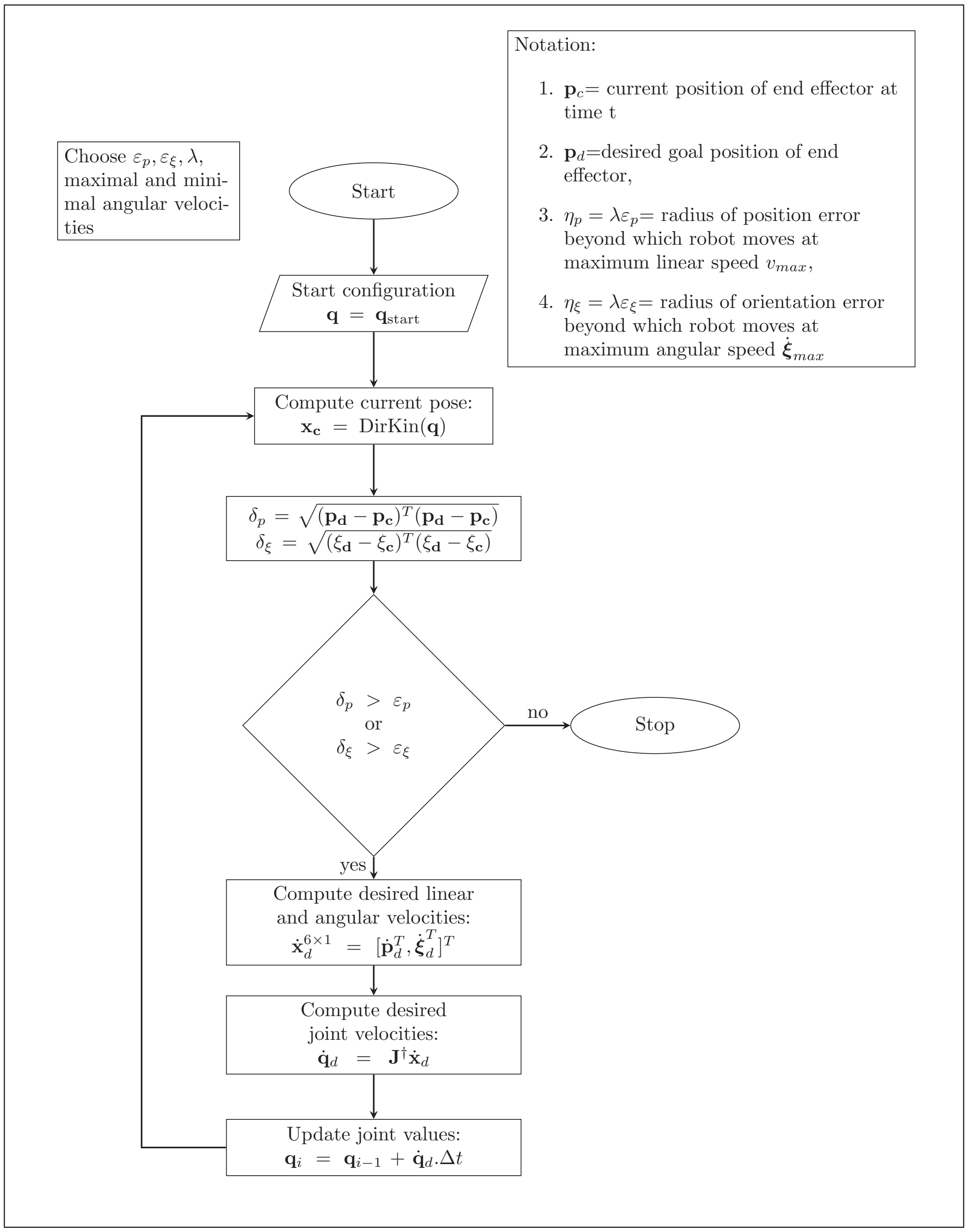

Notation:
1. $\mathbf{p}_c$= current position of end effector at time t
2. $\mathbf{p}_d$=desired goal position of end effector,
3. $\eta_p = \lambda\varepsilon_p$= radius of position error beyond which robot moves at maximum linear speed v_{max},
4. $\eta_\xi = \lambda\varepsilon_\xi$= radius of orientation error beyond which robot moves at maximum angular speed $\dot{\boldsymbol{\xi}}_{max}$
Choose $\varepsilon_p, \varepsilon_\xi, \lambda$, maximal and minimal angular velocities
Start
Start configuration $\mathbf{q} = \mathbf{q}_{\text{start}}$
Compute current pose: $\mathbf{x_c} = \text{DirKin}(\mathbf{q})$
$\delta_p = \sqrt{(\mathbf{p_d} - \mathbf{p_c})^T(\mathbf{p_d} - \mathbf{p_c})}$
$\delta_\xi = \sqrt{(\xi_\mathbf{d} - \xi_\mathbf{c})^T(\xi_\mathbf{d} - \xi_\mathbf{c})}$
$\delta_p > \varepsilon_p$ or $\delta_\xi > \varepsilon_\xi$
no
Stop
yes
Compute desired linear and angular velocities: $\dot{\mathbf{x}}_d^{6\times1} = [\dot{\mathbf{p}}_d^T, \dot{\boldsymbol{\xi}}_d^T]^T$
Compute desired joint velocities: $\dot{\mathbf{q}}_d = \mathbf{J}^\dagger\dot{\mathbf{x}}_d$
Update joint values: $\mathbf{q}_i = \mathbf{q}_{i-1} + \dot{\mathbf{q}}_d.\Delta t$

Figure II.3: Flowchart for the Resolved Rates Algorithm

II.3 Projections as a tool for virtual fixture specification

This section introduces the theory and mathematical background behind projections. A more rigorous definition of these concepts can be found in Basilevsky [23].

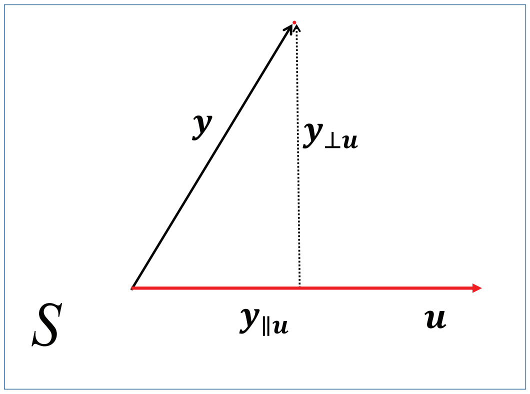


Figure II.4: The orthogonal projection of vector $\mathbf{y}$ onto vector $\mathbf{u}$

Whenever we define an n-dimensional vector $\mathbf{y} \in \mathbb{R}^n$, the coordinates of the vector represents the lengths of the orthogonal projection vectors of $\mathbf{y}$ onto n-coordinate basis vectors. Let $\mathbf{S}$ be defined as a r-dimensional subspace of vector space $\mathbf{V}$ where $(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \ldots, \mathbf{x_r})$ forms the basis for $\mathbf{S}$.

Consider the problem of projecting $\mathbf{y}$ onto subspace $\mathbf{S}$. In particular, if we consider $r = 1$, then this is a problem of projecting one vector $\mathbf{y} \in \mathbb{R}$ on another vector say $\mathbf{u} \in \mathbb{R}^n$. We have to decompose $\mathbf{y}$ into the sum of two components, one a multiple of non-zero vector $\mathbf{u}$ and the other orthogonal to $\mathbf{u}$ as shown in Figure II.4.

Therefore, we can express vector $\mathbf{y}$ as:

$$\mathbf{y} = \mathbf{y}_{\|\mathbf{u}} + \mathbf{y}_{\perp\mathbf{u}} \tag{II.9}$$

We can express the component along vector $\mathbf{u}$ as $\beta\mathbf{u}$ and the component perpendicular to $\mathbf{u}$ as $\mathbf{e}$ which would give the following,

$$\mathbf{y} = \beta\mathbf{u} + \mathbf{e}$$

where β is any scalar and $\mathbf{e}$ is orthogonal to $\mathbf{u}$. Therefore, we can write the following equation:

$$0 = \mathbf{e}.\mathbf{u} = (\mathbf{y} - \beta\mathbf{u}).\mathbf{u} = \mathbf{y}.\mathbf{u} - \beta(\mathbf{u}.\mathbf{u}) \tag{II.10}$$

From equation II.10 we can obtain the scalar β as,

$$\beta = \frac{\mathbf{y}.\mathbf{u}}{\mathbf{u}.\mathbf{u}}$$

We can finally define the projection of $\mathbf{y}$ on $\mathbf{u}$ as,

$$\mathbf{y}_{\|\mathbf{u}} = \frac{\mathbf{y}.\mathbf{u}}{\mathbf{u}.\mathbf{u}}.\mathbf{u} \tag{II.11}$$

Orthogonal Projection Matrices

In the earlier section, we considered a 1-dimensional subspace which was the simple case of projecting one vector on another. Now, we extend our approach to an r-dimensional subspace $\mathbf{S}$ given an n-dimensional vector $\mathbf{y}$ where $r < n$. The task would be to project the vector $\mathbf{y}$ onto the subspace $\mathbf{S}$ using a Projection Matrix $\mathbf{P}_A$.

Considering the case of an orthogonal projection as shown in Figure II.5 where we have have a subspace $\mathbf{S}$ defined by two basis vectors $(\mathbf{a_1}, \mathbf{a_2})$ and a 3-dimensional

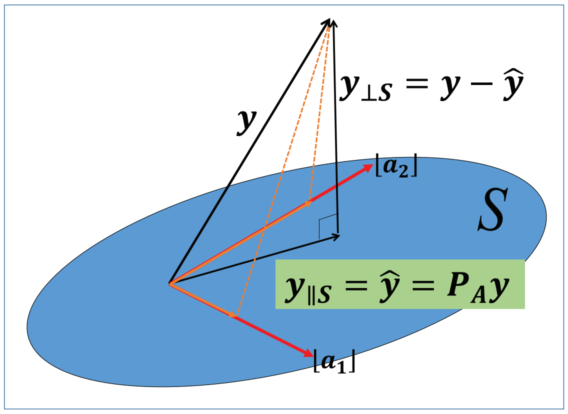

$y_{\perp S} = y - \hat{y}$
y
$[a_2]$
S
$y_{\parallel S} = \hat{y} = P_A y$
$[a_1]$

Figure II.5: The orthogonal projection of vector $\mathbf{y}$ onto a 2-dimensional subspace $\mathbf{S}$

vector $\mathbf{y} \in \mathbb{R}^3$. The vector $\mathbf{y}_{\|\mathbf{S}}$ can defined as,

$$\mathbf{y}_{\|\mathbf{S}} = \hat{\mathbf{y}} = \mathbf{P_A}\mathbf{y} \tag{II.12}$$

where $\mathbf{P_A}$ is the orthogonal projection matrix that projects a 3-dimensional vector $\mathbf{y}$ onto the 2-dimensional subspace $\mathbf{S}$. The vector $\mathbf{y}_{\perp\mathbf{S}}$ can be defined as,

$$\mathbf{y}_{\perp\mathbf{S}} = \mathbf{y} - \hat{\mathbf{y}} \tag{II.13}$$

Since this is an orthogonal projection, the vector $(\mathbf{y} - \hat{\mathbf{y}})$ is normal to the 2-dimensional subspace $\mathbf{S}$ which means that it is also orthogonal to its basis vectors $(\mathbf{a_1}, \mathbf{a_2})$. This gives us the following equations,

$$\mathbf{a_1}^T(\mathbf{y} - \hat{\mathbf{y}}) = \mathbf{a_1}^T(\mathbf{y} - \hat{\mathbf{y}}) = \mathbf{a_1}^T(\mathbf{y} - [\mathbf{a_1}x_1 + \mathbf{a_2}x_2]) = 0 \tag{II.14}$$

$$\mathbf{a_2}^T(\mathbf{y} - \hat{\mathbf{y}}) = \mathbf{a_2}^T(\mathbf{y} - \hat{\mathbf{y}}) = \mathbf{a_2}^T(\mathbf{y} - [\mathbf{a_1}x_1 + \mathbf{a_2}x_2]) = 0 \tag{II.15}$$

Considering,

$$\mathbf{A} = [\mathbf{a_1}, \mathbf{a_2}]$$

and

$$\mathbf{x} = [x_1, x_2]^T$$

where (x_1, x_2) are the coordinates of the projected vector $\hat{\mathbf{y}}$ in subspace $\mathbf{S}$. We can rewrite Equations II.14 and II.15 as,

$$\mathbf{A^T}(\mathbf{y} - \mathbf{Ax}) = 0$$

or,

$$\mathbf{A^T A x} = \mathbf{A^T y}$$

which gives us,

$$\mathbf{x} = (\mathbf{A^T A})^{-1}\mathbf{A^T y}$$

Therefore, the projection $\hat{\mathbf{y}}$ of vector y on the 2-dimensional subspace $\mathbf{S}$ is given as,

$$\hat{\mathbf{y}} = \mathbf{Ax} = \mathbf{A}(\mathbf{A^T A})^{-1}\mathbf{A^T y} = \mathbf{P_A y}$$

where $\mathbf{P_A}$ is an orthogonal projection matrix that projects any 3-dimensional vector onto the 2-dimensional subspace S defined by the columns of matrix $\mathbf{A}$.

Generalizing, if we have an n-dimensional vector $\mathbf{y}$ and an r-dimensional subspace $\mathbf{S}$ where $r < n$ and $\mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \ldots, \mathbf{x_r}]$ where $(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \ldots, \mathbf{x_r})$ form the basis for $\mathbf{S}$, then the Projection matrix can be derived as,

$$\mathbf{P_X} = \mathbf{X}(\mathbf{X^T X})^{-1}\mathbf{X^T} \tag{II.16}$$

If $\mathbf{P}$ is a Projection matrix, then the following theorems must be satisfied as well [23],

Theorem 1. $\mathbf{P}$ *must be idempotent i.e.* $\mathbf{P} = \mathbf{P}^2$

Theorem 2. $\mathbf{P}$ *must be symmetric to obtain an orthogonal projection.*

Proof. Referring Figure II.5, we know that vectors $(\mathbf{y} - \hat{\mathbf{y}})$ and $\hat{\mathbf{y}}$ are orthogonal to each other. Therefore, we can write:

$$\begin{aligned}(\mathbf{y} - \hat{\mathbf{y}})^T\hat{\mathbf{y}} &= (\mathbf{y} - \mathbf{P}\mathbf{y})^T\mathbf{P}\mathbf{y} \\ &= [(\mathbf{I} - \mathbf{P})\mathbf{y}]^T\mathbf{P}\mathbf{y} \\ &= \mathbf{y}^T(\mathbf{I} - \mathbf{P})^T\mathbf{P}\mathbf{y} = 0 \qquad \text{(II.17)}\end{aligned}$$

Since we want II.17 to hold for every $\mathbf{y}$, we have

$$\begin{aligned}(\mathbf{I} - \mathbf{P})^T\mathbf{P} &= 0 \\ \mathbf{P}^T\mathbf{P} &= \mathbf{P} \qquad \text{(II.18)}\end{aligned}$$

Taking the transpose of II.18 gives:

$$\mathbf{P}\mathbf{P}^T = \mathbf{P}^T \qquad \text{(II.19)}$$

Equations II.18 and II.19 give us:

$$\mathbf{P}^T = \mathbf{P} \qquad \text{(II.20)}$$

Therefore, this proves Theorems 1 and 2. □

Theorem 3. *If* $\mathbf{X}$ *is an* $n \times k$ *matrix with rank* $k \times n$*, where the columns of* $\mathbf{X}$ *define the subspace* $\mathbf{S}$*, then* $\mathbf{P_X} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ *is idempotent and symmetric i.e. it is an orthogonal projection matrix.*

Proof of symmetry.

$$\begin{aligned}
\mathbf{P_X}^T &= [\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T]^T \\
&= \mathbf{X}[(\mathbf{X}^T\mathbf{X})^{-1}]^T\mathbf{X}^T \\
&= \mathbf{X}[(\mathbf{X}^T\mathbf{X})^T]^{-1}\mathbf{X}^T \\
&= \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T = \mathbf{P_X}
\end{aligned}$$

Since $\mathbf{P_X}^T = \mathbf{P_X}$, $\mathbf{P_X}$ is symmetric. □

Proof of idempotency.

$$\begin{aligned}
\mathbf{P_X}^2 &= \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T \\
&= \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T = \mathbf{P}_X
\end{aligned}$$

□

Theorem 4. $\mathbf{I} - \mathbf{P}_X$ *spans the nullspace of* $\mathbf{X}$

Proof. To prove that $\mathbf{I} - \mathbf{P}_X$ spans the nullspace of $\mathbf{X}$, we have to show that $I - P_X$ projects a vector in range($\mathbf{X}$) to the null vector $\mathbf{0}$.

$$\begin{aligned}
(\mathbf{I} - \mathbf{P_X})\mathbf{X}\mathbf{y} &= \mathbf{X}\mathbf{y} - \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}\mathbf{y} \\
&= \mathbf{0}
\end{aligned}$$

□

Oblique Projection Matrices

We know that the Projection matrix $\mathbf{P}$ is orthogonal *iff* the matrix is idempotent and symmetric. We can generalize the projection matrix $\mathbf{P}$ by transforming a vector $\mathbf{y}$ onto a different projection vector $\hat{\mathbf{y}}$ such that the vector $\mathbf{y} - \hat{\mathbf{y}}$ is not normal to the subspace $\mathbf{S}$ (refer Figure II.5)

Let $\boldsymbol{\Phi}$ be a positive definite $n \times n$ matrix. It can be proven that there exists a non-singular $n \times n$ matrix $\mathbf{C}$ such that $\mathbf{C}\boldsymbol{\Phi}\mathbf{C}^T = \mathbf{I}$ and $\mathbf{C}^T\mathbf{C} = \boldsymbol{\Phi}^{-1}$, so that $\boldsymbol{\Phi}$ is symmetric [23].

If we have an r-dimensional subspace $\mathbf{S}$ where $r < n$ and $\mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \ldots, \mathbf{x_r}]$ where $(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \ldots, \mathbf{x_r})$ form the basis for $\mathbf{S}$, we can transform the matrix $\mathbf{X}$ as $\mathbf{X}^* = \mathbf{C}\mathbf{X}$ [23]. Substituting this in the equation for a Projection matrix, we would get,

$$\begin{aligned}
\mathbf{P}_{\mathbf{X}^*} &= \mathbf{X}^*(\mathbf{X}^{*T}\mathbf{X}^*)^{-1}\mathbf{X}^{*T} \\
&= \mathbf{C}\mathbf{X}(\mathbf{X}^T\mathbf{C}^T\mathbf{C}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{C}^T \\
&= \mathbf{C}\mathbf{X}(\mathbf{X}^T\boldsymbol{\Phi}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{C}^T
\end{aligned}$$

Now, if we consider $\mathbf{y}^* = \mathbf{C}\mathbf{y}$, then the projection $\hat{\mathbf{y}}^* = \mathbf{P}_{\mathbf{X}^*}\mathbf{y}^*$ would give us the following,

$$\begin{aligned}
\hat{\mathbf{y}}^* &= (\mathbf{C}\mathbf{X}(\mathbf{X}^T\boldsymbol{\Phi}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{C}^T)\mathbf{y}^* \\
\mathbf{C}^{-1}\hat{\mathbf{y}}^* &= (\mathbf{X}(\mathbf{X}^T\boldsymbol{\Phi}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{\Phi}^{-1})\mathbf{y} \\
\hat{\mathbf{y}} &= \mathbf{P}_{\mathbf{X}}\mathbf{y}
\end{aligned}$$

Therefore, the oblique projection matrix in terms of the original axes is given by,

$$\mathbf{P}_{\mathbf{X}} = \mathbf{X}(\mathbf{X}^T\mathbf{\Phi}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{\Phi}^{-1} \tag{II.21}$$

II.4 Kinematic filtering for Virtual Fixtures

As discussed before in the literature review section, there are two basic approaches to implementing virtual fixtures. We can implement them as "barrier virtual fixtures" which prevents erroneous tool excursions by the user. They can also be implemented as assistive tools which guide the user in following a curve or a plane. This approach is termed as *"guidance virtual fixtures"*. In this section, we would discuss the implementation of *"guidance virtual fixtures"* using projections, where virtual fixtures are treated as geometrical constraints [21, 24].

In our case, we have an admittance type robot which is either controlled cooperatively through the use of a force sensor attached to the robot end effector or telemanipulated through a master device. In the case with cooperative manipulation, the user directs the tool attached to the force sensor. The detected forces $\mathbf{f} = (f_x, f_y, f_z)^T$ are expressed in the robot base frame of reference. These forces are multiplied with a gain term to get the commanded master velocity $\dot{\mathbf{x}}_m$. In telemanipulation, we compute the commanded master velocity $\dot{\mathbf{x}}_m$ through the relative motion of the master device with respect to a set anchor point.

Our task is to filter the commanded velocity $\dot{\mathbf{x}}_m$ into allowable and constrained directions such that we can constrain the robot to move along specified geometric features. We would derive the virtual fixture equations for three cases,

1. Movement along a plane
2. Movement along a Curve
3. Staying within a Curve and on a Plane

Using Projections to implement VF

The basic idea behind virtual fixtures is that we have to decompose the commanded velocity $\dot{\mathbf{x}}_m$ along the allowable $\dot{\mathbf{x}}_{m_{des}}$ and constrained $\dot{\mathbf{x}}_{m_\tau}$ directions. We then use these components to get an equation for the desired slave velocity $\dot{\mathbf{x}}_d$.

So, if we have a k-dimensional subspace $\mathbf{S}$ of allowable directions defined by its basis vectors $[\mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_k}]$, we can derive the Projection matrix $\mathbf{P}$ using the following equation,

$$\begin{aligned} &\mathbf{P} = \mathbf{X}(\mathbf{X^T X})^{-1}\mathbf{X^T} \\ &\quad \text{where } \mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \dots, \mathbf{x_r}] \end{aligned} \tag{II.22}$$

We know that the Projection matrix $\mathbf{P}$ projects a vector $\mathbf{y} \in \mathbb{R}$ orthogonally into its subspace *iff* $\mathbf{P}$ is idempotent and symmetric (refer to Theorem 3). Also, we know that $(\mathbf{I} - \mathbf{P})$ spans the nullspace of $\mathbf{X}$. Therefore, if we want to decompose the vector $\dot{\mathbf{x}}_m$ into its allowable and forbidden components, we use the following equations,

$$\begin{aligned} \text{Allowable component } &= \dot{\mathbf{x}}_{m_{des}} = \mathbf{P}\mathbf{x}_m \\ \text{Forbidden component } &= \dot{\mathbf{x}}_{m_\tau} = (\underbrace{\mathbf{I} - \mathbf{P}}_{\tilde{\mathbf{P}}})\dot{\mathbf{x}}_m = \tilde{\mathbf{P}}\dot{\mathbf{x}}_m \end{aligned} \tag{II.23}$$

where $\mathbf{P}$ and $\mathbf{X} \in \mathbb{R}^{6\times n}$ $(n < 6)$, contain columns of allowable(preferred) movement directions. Next, we derive the kinematic filtering equations for implementing virtual fixtures for each of the three cases.

Virtual Fixtures constraining movement to a plane

We know that a plane can be described by two vectors that lie on the plane. As shown in Figure II.6, the plane is described by two vectors $(\mathbf{x}_1, \mathbf{x}_2)$. The plane is considered as a 2-dimensional subspace $\mathbf{S}$ with its allowable directions defined by matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2]$. The Projection matrix for the plane $\mathbf{P}_\pi$ is defined using Equation II.16 as,

$$\mathbf{P}_\pi = \mathbf{X}(\mathbf{X}^\mathbf{T}\mathbf{X})^{-1}\mathbf{X}^\mathbf{T} \tag{II.24}$$

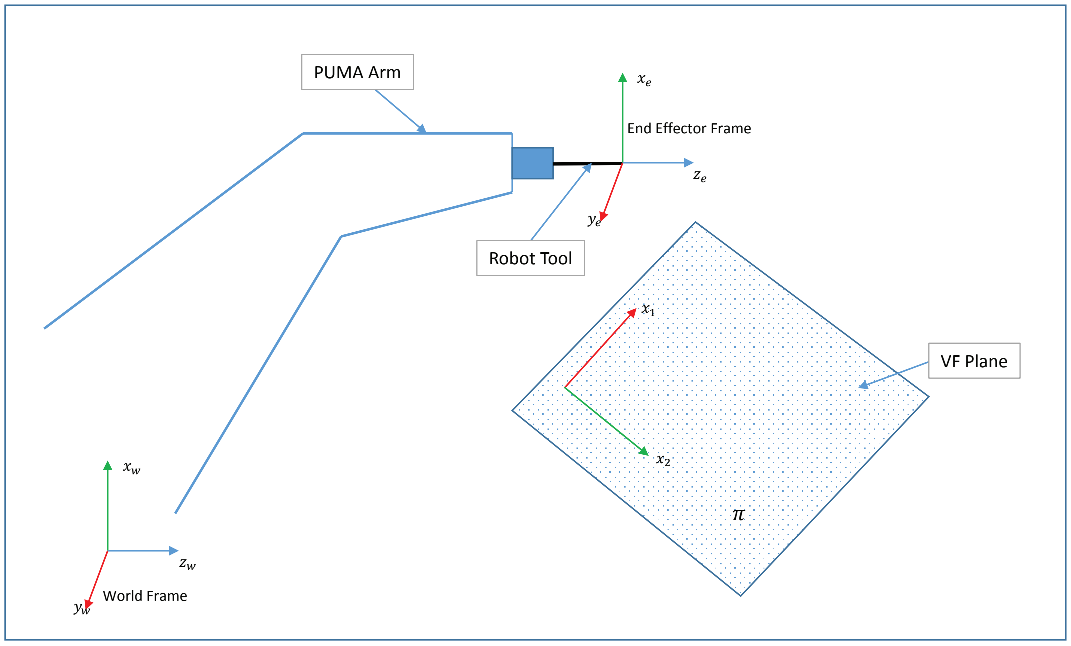


Figure II.6: Virtual Fixtures constraining movement to a Plane

Using $\mathbf{P}_\pi$, we can decompose the commanded twist into its allowable $\dot{\mathbf{x}}_{m_{des}}$ and forbidden $\dot{\mathbf{x}}_{m_\tau}$ components (refer Equation II.23) such that,

$$\dot{\mathbf{x}}_m = \underbrace{\mathbf{P}_\pi \dot{\mathbf{x}}_m}_{\dot{\mathbf{x}}_{m_{des}}} + \underbrace{\tilde{\mathbf{P}}_\pi \dot{\mathbf{x}}_m}_{\dot{\mathbf{x}}_{m_\tau}} \tag{II.25}$$

We can use a simple admittance law to to admit motions in the allowable directions as follows,

$$\begin{aligned} \dot{\mathbf{x}}_{des} &= K_{a_1} \dot{\mathbf{x}}_{m_{des}} \\ &= K_{a_1} \mathbf{P}_\pi \dot{\mathbf{x}}_m \\ &\text{where } K_{a_1} \text{is the admittance gain} \end{aligned} \tag{II.26}$$

Equation II.26 provides a hard constraint against moving in forbidden directions. The issue with implementing a hard constraint is that it requires perfect knowledge of the task geometry and allows no deviation from the task. It also provides a sudden “hard stop” when the user reaches the constraint boundary. This equation would result in a jerky feeling close to the boundary unless the admittance gain is adaptively adjusted. If we want to let the user deviate a little bit in the forbidden directions, we need a soft constraint admittance law using $\dot{\mathbf{x}}_{m_\tau}$ as an additional input,

$$\begin{aligned} \dot{\mathbf{x}}_{des} &= K_{a_2} \dot{\mathbf{x}}_{m_\tau} \\ &= K_{a_2} \tilde{\mathbf{P}}_\pi \dot{\mathbf{x}}_m, \quad K_{a_2} << K_{a_1} \\ &\text{where } K_{a_2} \quad \text{is the admittance gain} \end{aligned} \tag{II.27}$$

Combining Equations II.26 and II.27, we get the following equation:

$$\dot{\mathbf{x}}_{des} = K_{a_1}\mathbf{P}_\pi\dot{\mathbf{x}}_m + K_{a_2}\tilde{\mathbf{P}}_\pi\dot{\mathbf{x}}_m \tag{II.28}$$

where K_{a_1} and K_{a_2} are the two admittance gains that adjust the motion responsiveness. We can also write Equation II.28 as:

$$\begin{aligned}\dot{\mathbf{x}}_d &= K_a(\dot{\mathbf{x}}_{m_{des}} + K_\tau\dot{\mathbf{x}}_{m_\tau}) \\ &= K_a(\mathbf{P}_\pi + K_\tau\tilde{\mathbf{P}}_\pi)\dot{\mathbf{x}}_m \end{aligned}\tag{II.29}$$

Now, Equation II.29 would still cause errors if we want to constrain the motion to a plane. The reason is that integration errors cause a drift of the robot end effector from the intended motion plane. The end effector still moves parallel to the constraint plane, however the controller has no ability to overcome this drift. We need a term that closes this error when the commanded velocity $\dot{\mathbf{x}}_m$ is zero. To compensate for the

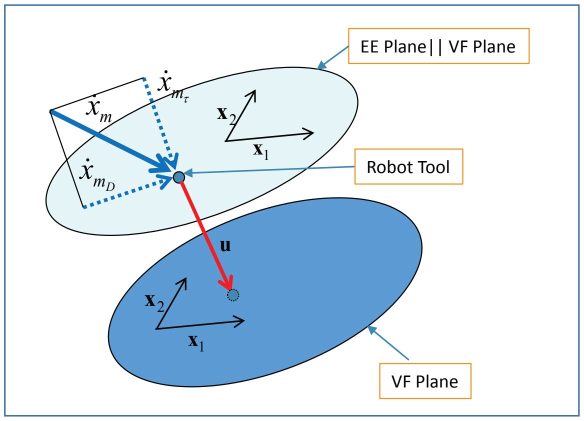


Figure II.7: Vector $\mathbf{u}$ closing error to the Plane π

drift, we consider a unit vector $\hat{\mathbf{u}}$ that points from the current end effector position to the closest point on the constraint plane which is the normal. Using vector $\hat{\mathbf{u}}$, we add a corrective term that brings the end effector back to the Plane π to get the final control equation as follows,

$$\dot{\mathbf{x}}_d = K_a(\mathbf{P}_\pi + K_\tau \tilde{\mathbf{P}}_\pi)\dot{\mathbf{x}}_m + K_{p_\pi}\hat{\mathbf{u}} \tag{II.30}$$

Equation II.30 always brings the end effector back to the plane. In the current implementation, we trust the robot kinematics to be accurate. However, if we have visual feedback, then vector $\hat{\mathbf{u}}$ can be computed using vision which could improve accuracy.

Virtual Fixtures constraining movement along a curve

The approach for deriving the VF control equation to follow a curve is very similar to the previous case. As shown in Figure II.8, the task is to follow a curve $\mathbf{c}$ on a plane π. To constrain the movement of the end effector along a curve, we would need to make sure that the computed slave twist $\dot{\mathbf{x}}_d$ lies along the curve tangent and on the plane. Therefore, we would need to derive the projection matrix for the curve as well as the plane. The projection matrix for the plane $\mathbf{P}_\pi$ can be computed using equation II.24. The allowable directions of motion on the plane π are defined by the matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2]$ where $(\mathbf{x}_1, \mathbf{x}_2)$ define the plane.

$$\mathbf{P}_\pi = \mathbf{X}(\mathbf{X}^\mathbf{T}\mathbf{X})^{-1}\mathbf{X}^\mathbf{T}$$

To compute the projection matrix for the curve, we need to compute the local tangent $\hat{\mathbf{t}}$ to the closest point on the curve $\mathbf{c}$. When the curve is given as a set of points, we

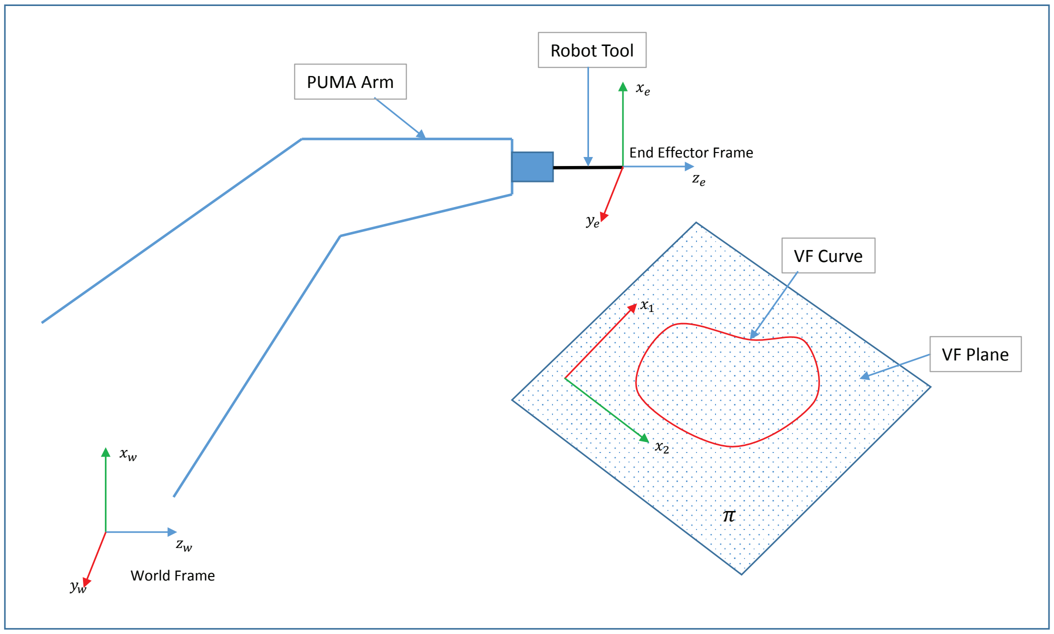


Figure II.8: Virtual fixture constraining movement along a curve c

can compute the numerical approximation of the local tangent. Once we have the local tangent at the closest point on the curve, we can compute the projection matrix for the curve as follows,

$$\mathbf{P}_c = \hat{\mathbf{t}}(\hat{\mathbf{t}}^T\hat{\mathbf{t}})^{-1}\hat{\mathbf{t}}^T$$

Just like the previous case, we have to project the commanded twist on plane π and curve c. The allowable projection, has to be along the curve and on the plane. Therefore, we have to make sure that the allowable velocity component lies along the curve and on the plane. However, projecting along the curve would ensure that we are on the plane as well. Therefore, we just need the projection matrix for curve c.

For the constrained direction, we can move away from the curve but we would still like to move along the plane π. Therefore, we can use the nullspace of projection matrix $\mathbf{P}_c$ to compute the twist in forbidden directions. However, to make sure that we still lie on the plane, we have to project this twist along the plane as well using $\mathbf{P}_\pi$. Therefore, we can write the equations as follows,

$$\begin{aligned} \text{Allowable component } &= \dot{\mathbf{x}}_{m_{des}} = \mathbf{P}_c\mathbf{P}_\pi\mathbf{x}_m = \mathbf{P}_c\mathbf{x}_m \\ \text{Forbidden component } &= \dot{\mathbf{x}}_{m_\tau} = \underbrace{(\mathbf{I}-\mathbf{P}_c)}_{\tilde{\mathbf{P}}_c}\mathbf{P}_\pi\dot{\mathbf{x}}_m = \tilde{\mathbf{P}}_c\mathbf{P}_\pi\dot{\mathbf{x}}_m \end{aligned} \quad \text{(II.31)}$$

Again, Equation II.31 does not account for integration errors. In this case we have to compensate for the drift from the plane as well as the curve.

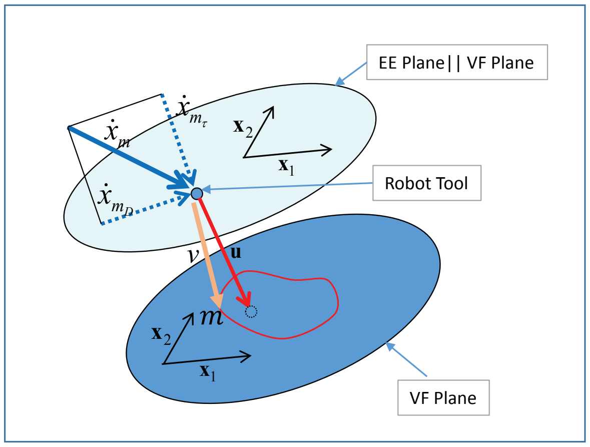


Figure II.9: Vectors $\mathbf{u}$ and $\mathbf{v}$ closing errors to the Plane π and curve c

To compensate for the drift, we would again have a unit vector $\hat{\mathbf{u}}$ pointing to the closest point on the constraint plane which is the normal to the plane. Also, we would have a unit vector $\hat{\mathbf{v}}$ pointing to the closest point on the curve as shown in Figure II.9. Using vectors $\mathbf{u}$ and $\mathbf{v}$, we add corrective terms that brings the end effector back to the Plane π and curve c.

$$\dot{\mathbf{x}}_d = K_a(\mathbf{P}_c + K_\tau \tilde{\mathbf{P}}_c \mathbf{P}_\pi)\dot{\mathbf{x}}_m + K_{p_\pi}\hat{\mathbf{u}} + K_{p_c}\hat{\mathbf{v}} \tag{II.32}$$

Equation II.32 would compensate for the drift from the plane and curve using the proportional gains K_{p_π} and K_{p_c} along $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ respectively.

Virtual Fixtures constraining movement within a curve

As shown in Figure II.10, the task is to stay within a curve **c** on a plane π. To constrain the movement within the curve, we would just need to modify Equation II.32. We know that if we set the proportional constraint gain K_τ to 1, we get isotropic admittance whereas if it is set to 0 we get a hard constraint. We would just need to

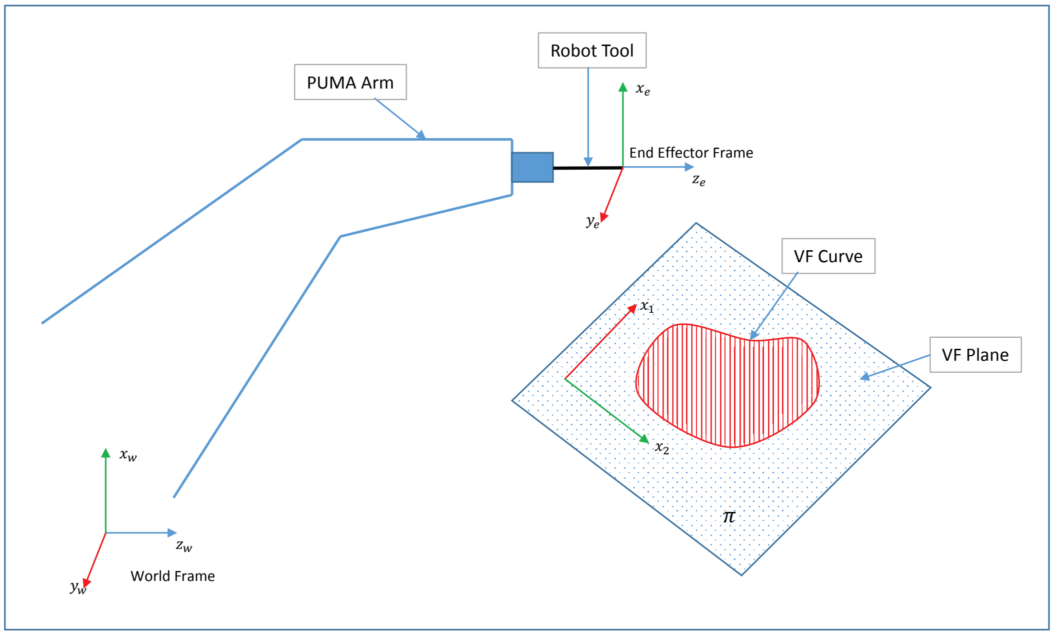


Figure II.10: Virtual Fixtures constraining movement within a curve c

modify the proportional constraint gain K_τ such that we have isotropic admittance when the end effector of the robot is within the curve. However, as we start moving towards the boundary of the curve the gain K_τ should start decreasing until it reaches 0 close to the curve boundary. We call this varying K_τ as $K_{\tau_{computed}}$ in our modified equation

$$\dot{\mathbf{x}}_d = K_a(\mathbf{P}_c + K_{\tau_{computed}}\tilde{\mathbf{P}}_c\mathbf{P}_\pi)\dot{\mathbf{x}}_m + K_{p_\pi}\hat{\mathbf{u}} + K_{p_c}\hat{\mathbf{v}} \tag{II.33}$$

To make $K_{\tau_{computed}}$ vary smoothly, we make use of the hyperbolic tangent function as follows,

$$K_{\tau_{computed}} = \frac{1}{2}[1 + \tanh(p(x-a))] \tag{II.34}$$

where p is the proportional gain and a is the offset from origin where the transition between 0 and 1 occurs. This equation gives us a smooth varying value of $K_{\tau_{computed}}$ as shown in Figure II.11

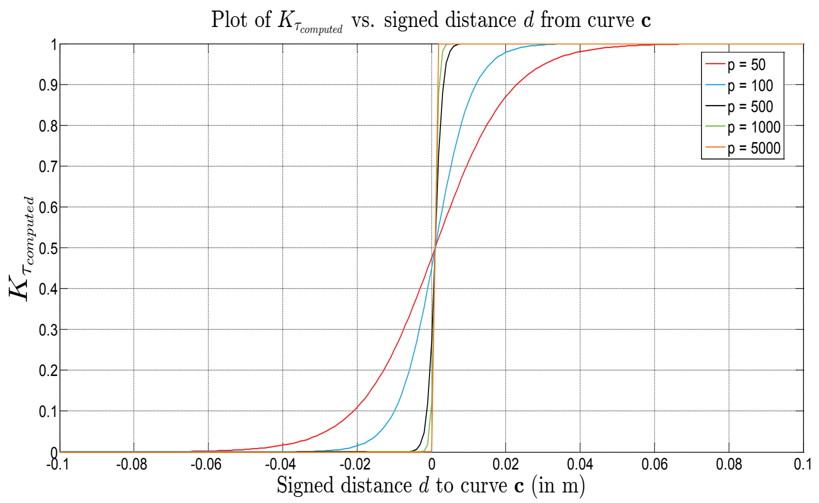


Figure II.11: Plots of $K_{\tau_{computed}}$ for different proportional gains

As we can see from Figure II.11, the gain $K_{\tau_{computed}}$ is equal to 1, when we are inside the curve, but as we start moving towards the curve $K_{\tau_{computed}}$ starts reducing gradually until it reaches 0, close to the boundary of the curve. We observe that varying the proportional gain term p in equation II.34 varies the slope of the transition at the curve boundary. A higher value causes a sharp transition between 1 and 0. The value of a in equation II.34 is chosen to be $1/10^{\text{th}}$ the radius of the curve. This ensures that as we reach the curve boundary the value of $K_{\tau_{computed}}$ is close to 0. During experimental validation, we have to choose a value of p that gives us the necessary transition characteristic.

CHAPTER III

USER INTERFACE FOR REAL-TIME CONTROL OF THE PUMA 560 ROBOT

This chapter presents the implementation of the user interface developed for the real time control of the PUMA robot. The real-time control framework for independently operable control modes was developed by ARMA members Andrea Bajo, Long Wang and Jason Pile. This work focused on generating a control code that integrates all these operation modes while offering seamless transition between the modes. The user interface developed integrates the various modes of operation of the PUMA robot ensuring safe transition between the modes.

III.1 PUMA Kinematic model

Before we discuss the robot control code, let us first discuss the kinematic model of the PUMA robot. Figure III.1 shows the definitions of the frames using the modified Denavit-Hartenberg convention [25].

The frames are assigned according to the DH convention which is described in detail in [25, 26]. Briefly, the z-axis for each joint is placed along its axis of motion, and then the x-axes are placed along the common normals between these z-axes. The y-axes are chosen to satisfy the right-handed coordinate system.

1. Staring from the base frame, the z_0 axis is chosen along the first joint axis and x_0 selected to be parallel to the second link.

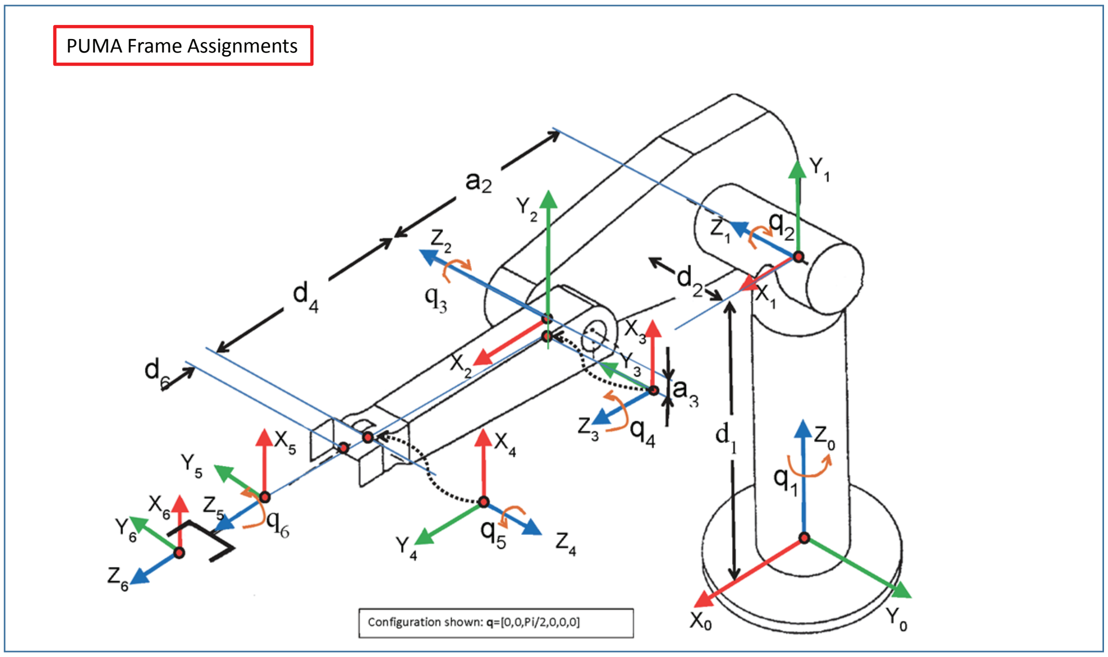


Figure III.1: PUMA 560 frame assignments

2. The z_1 axis is along the second joint axis, the origin being at the point of intersection between z_0 and z_1. x_1 is free for us to choose in this case and it is chosen along the second link.

3. The z_2 axis is chosen along the third joint axis and is parallel to z_1. The x_2 axis is chosen along the common normal of z_1 and z_2.

4. The origin of Frame 3 is chosen to be above Frame 2. The z_3 axis points along the fourth joint axis and x_3 along the common normal of z_2 and z_3.

5. The z_4 axis is chosen along the fifth joint axis and x_4 is free for us to choose and is chosen along x_3

6. The z_5 axis is chosen along the sixth joint axis and x_5 is again free for us to choose and is chosen along x_3.

7. Frame 6 is the end effector frame to complete the DH table, where all three axes are coincident with the fifth frame

LINK	θ_i	d_i (m)	a_i (m)	α_i(radians)
1	q_1	0.6718	0	$\pi/2$
2	q_2	0.1501	0.4318	0
3	q_3	0	-0.0203	$\pi/2$
4	q_4	0.4331	0	$\pi/2$
5	q_5	0	0	$-\pi/2$
6	q_6	0.0558	0	0

Table III.1: DH parameters for the Puma robot assuming the operational point is at the center of the end effector flange

Let us define the DH parameters briefly,

1. θ_i is the angle by which x_{i-1} rotates about z_{i-1} axis to come into alignment with x_i according to the right-hand rule. It is a variable for a revolute joint and a constant for a prismatic joint.

2. d_i is the distance between x_{i-1} axis and x_i axis measured along z_{i-1}. It is a constant for a revolute joint and a variable for a prismatic joint.

3. a_i is the distance between z_{i-1} and z_i measured along x_i. a_i is a constant.

4. α_i is the angle required to rotate the z_{i-1} axis into alignment with the z_i axis according to the right-hand rule.

The homogeneous transform from frame $i-1$ to frame i is given by the following equation:

$$\mathbf{T}_i^{i-1} = \begin{bmatrix} c_{\theta_i} & -c_{\alpha_i}s_{\theta_i} & s_{\alpha_i}s_{\theta_i} & a_ic_{\theta_i} \\ s_{\theta_i} & c_{\alpha_i}c_{\theta_i} & -s_{\alpha_i}c_{\theta_i} & a_is_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{III.1}$$

where $c_x = \cos(x)$ and $s_x = \sin(x)$

Thus, the direct kinematics of the PUMA robot can be computed as:

$$\mathbf{T} = \mathbf{T}_6^0 = \prod_{i=1}^{6} \mathbf{T}_i^{i-1} \tag{III.2}$$

III.2 Real-time Control using MATLAB® xPC

The PUMA robot control code was implemented using xPC Target which is a real-time software environment provided with MATLAB®. The advantage with using xPC target is that we can use Simulink and stateflow models to build up the control system which allows for rapid testing of control algorithms on the physical hardware.

The xPC Target operating system executes on a target machine which runs the control system for the robot in real-time and is interfaced with the robot through D/A converters and servo amplifiers. The robot encoders and potentiometers are connected to DAC Cards which read in the signals and sends analog signals computed by the controller to the servo amplifiers which drive the robot motors. The target machine also communicates with a host system through a Local Area Network or through a direct serial connection. We can observe and vary the controller parameters through the host system by communicating with the target machine.

The PUMA Controller comprises of three major subsystems as shown in Figure III.2,

1. Trajectory Planner
2. PD + Inverse Dynamics
3. Robot Controller Interface

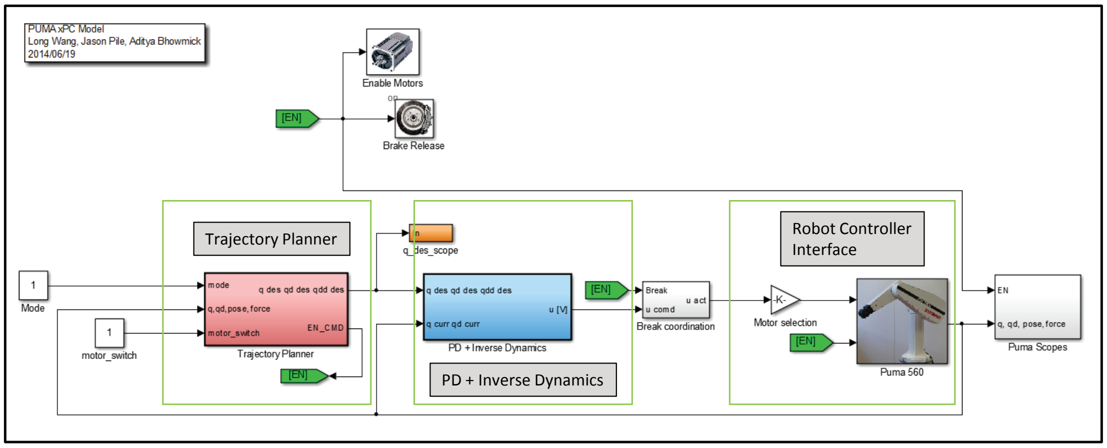


Figure III.2: xPC model of the PUMA Controller

The implementation of the PUMA controller was done by ARMA members (A. Bajo, Long Wang, Jason Pile). We briefly describe each subsystem in the following sections.

PD + Inverse Dynamics Controller

The *PD + Inverse Dynamics* subsystem implements the control equation of the robot (Figure III.3). The code for this subsystem was written by Andrea Bajo.

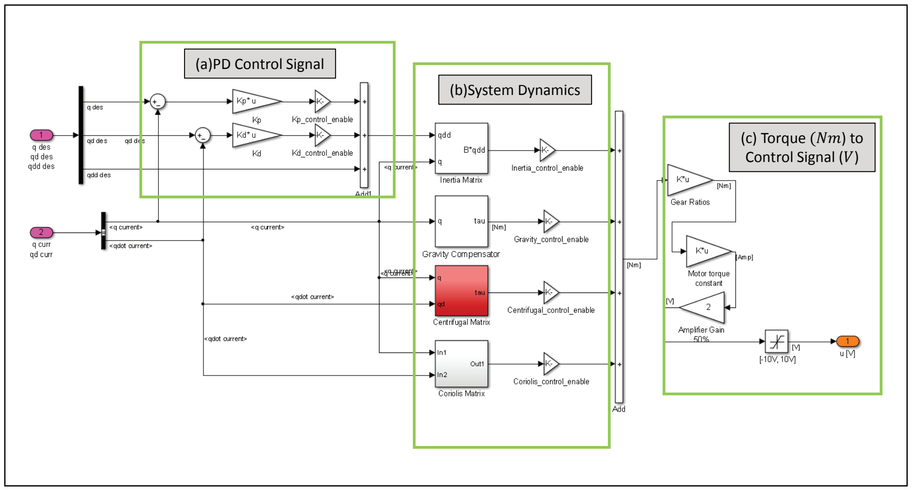


Figure III.3: PD + Inverse Dynamics subsystem

The desired values of joint positions $\mathbf{q}_d$ and velocities $\dot{\mathbf{q}}_d$ (along with the desired joint acceleration $\dot{\mathbf{q}}_d$) are compared with the current values of joint positions $\mathbf{q}_c$ and velocities $\dot{\mathbf{q}}_c$ to generate the error vector $\mathbf{e}$ and its derivative $\dot{\mathbf{e}}$. These error vectors are multiplied by the gain matrices $\mathbf{K}_P$ and $\mathbf{K}_D$. The output signal is then fed into the non-linear inner loop of the controller which comprises of the Gravity compensator, and the Inertial, Centrifugal and Coriolis matrices. The output signal of the non-linear controller is a vector of desired joint torques $\boldsymbol{\tau}_d$ which is expressed in Nm. This signal is then converted into a voltage signal which is fed to the D/A converter as an input. The D/A converter then sends the analog voltage signal to the servo amplifiers powering the robot motors.

Robot Controller Interface

The *PUMA560* subsystem implements the robot controller interface as shown in Figure III.4. The code for this subsystem was written by Long Wang and Jason Pile. This subsystem controls the D/A converters which give the required control voltages to the servo amplifiers. There are four main components labelled in the figure. Figure

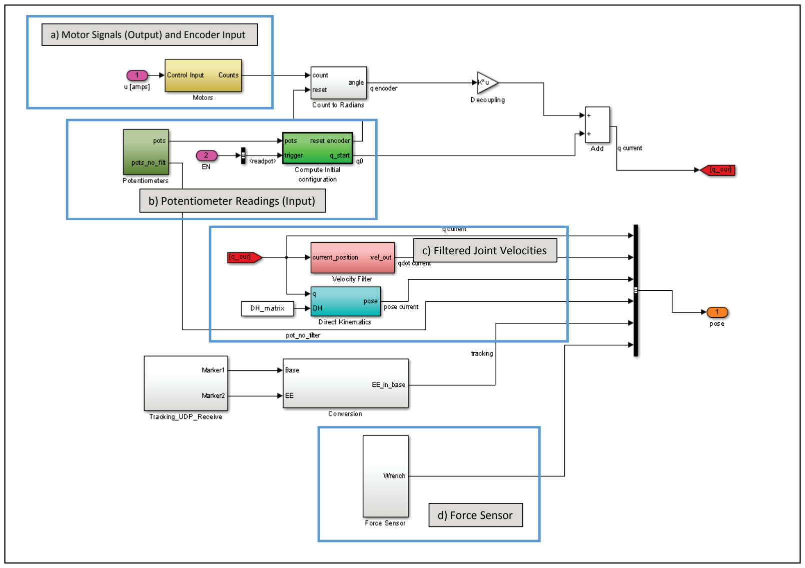


Figure III.4: Robot Controller Interface

III.4-a refers to the section which sends the motor signals to the D/A converters and also reads from the joint encoders. Figure III.4-b refers to the section which reads in the potentiometer values from the robot. Figure III.4-c refers to the section which filters the current joint velocities $\dot{\mathbf{x}}_c$. Figure III.4-d refers to the block which reads the force sensor attached to the PUMA end effector.

Trajectory Planner

The trajectory planner block consists of three main subsystems, the Initialization subsystem(Figure III.5-a), the Joint Space subsystem(Figure III.5-b), and the Task Space subsystem(Figure III.5-c). The code for this subsystem was written by Long Wang and Jason Pile. The *Initialization* subsystem is used for starting up the robot controller and takes care of initializing the joint values from the potentiometer readings. The *Joint Space* subsystem is used for joint space control which uses a

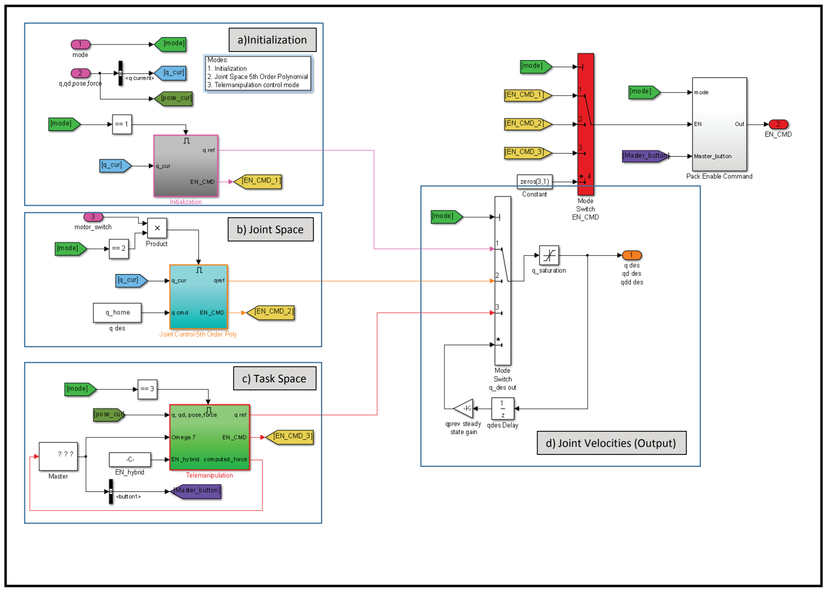


Figure III.5: Trajectory Planner

fifth order polynomial planner to ensure smooth operation of the PUMA robot. The *Task Space* subsystem is used for telemanipulation and cooperative manipulation of the PUMA robot. This subsystem contains the implementation of virtual fixtures presented in this thesis.

Application Specific Code

This subsystem is contained in the Task Space block of the Trajectory Planner and has the implementation of the hybrid admittance controller and the Virtual Fixtures subsystem. The first block computes the desired velocity of the PUMA robot. This desired velocity is represented as $\dot{\mathbf{x}}_m$ in our derivation of the kinematic filtering equations for virtual fixtures. The *Hybrid Admittance* block (Figure III.6-a) contains the implementation of the hybrid admittance controller which is used in cooperative manipulation. The *Virtual Fixture* block (Figure III.6-b) contains the

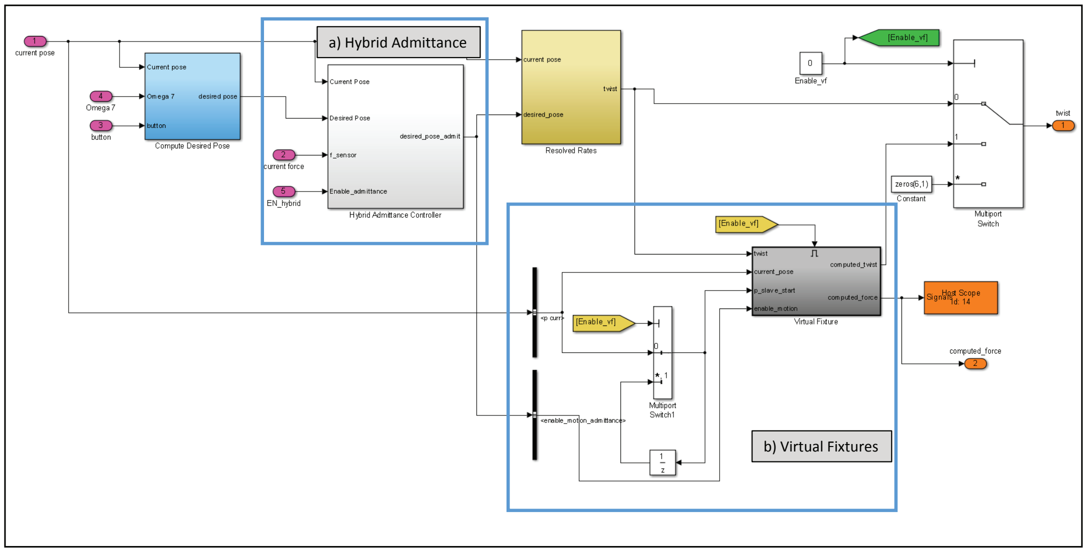


Figure III.6: Subsystem of Task Space block

implementation of virtual fixture laws. It filters the input velocity coming in from the Hybrid Admittance block and computes the desired velocity $\dot{\mathbf{x}}_d$ which is given to the resolved rates subsystem of the Task Space block.

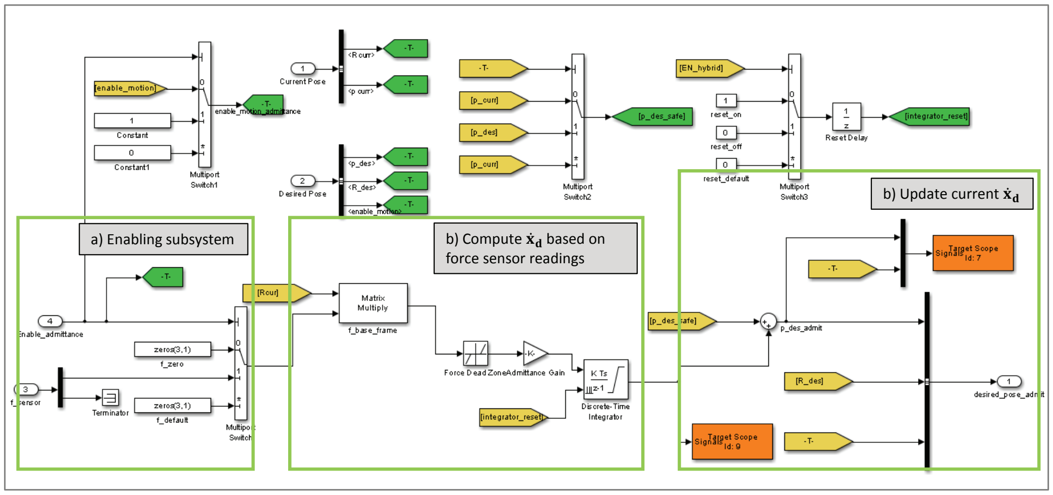


Figure III.7: Hybrid Admittance block

The *Hybrid Admittance* block is used for cooperative manipulation of the PUMA using the force sensor attached to the end effector. The initial version of this model was provided by Jason Pile. It consists of three main sections. The first section (Figure III.7-a) is used for enabling the hybrid admittance. The block only updates the desired velocity from the force sensor if the subsystem is enabled. The second section (Figure III.7-b) is responsible for updating the desired velocity using the force sensor readings $\mathbf{f} = (f_x, f_y, f_z)^T$ expressed in the robot base frame of reference. These forces are multiplied with a gain term to update the desired velocity $\dot{\mathbf{x}}_d$ (Figure III.7-c).

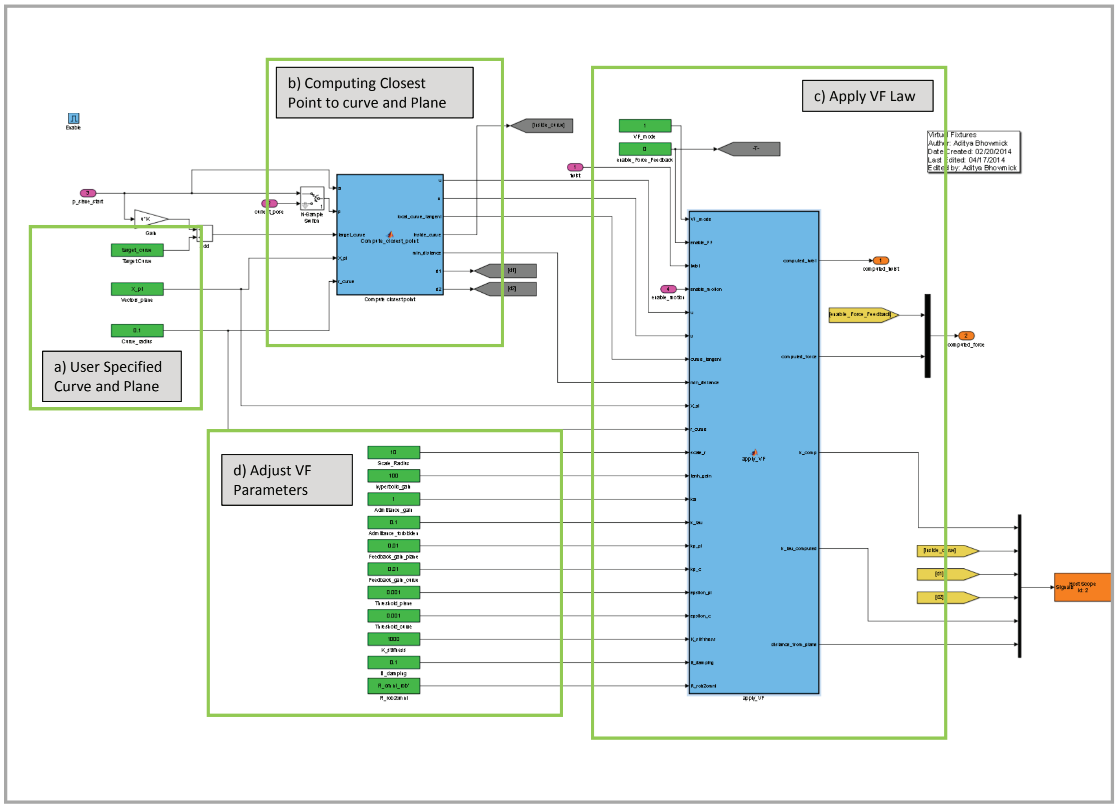


Figure III.8: Virtual Fixture block

The *Virtual Fixture* block (Figure III.8) is used for applying the virtual fixture laws to get the desired velocity $\dot{\mathbf{x}}_d$ which is geometrically constrained according to the selected virtual fixture law. The constant blocks in Figure III.8-a are used to specify the curve $\mathbf{c}$ and the plane $\boldsymbol{\pi}$ for applying virtual fixtures. The curve is represented by a set of points and the plane is represented by its two basis vectors. The function block in Figure III.8-b is used to compute the closest point to the curve and the plane. This data is fed into the next block Figure III.8-c which applies the virtual fixture laws. We can adjust the various parameters of the virtual fixture laws using the constant blocks in Figure III.8-d. These constant blocks can be modified in real-time from the graphical user interface. The graphical user interface was developed to integrate all the modes of operation of the PUMA robot and control the virtual fixture parameters of the robot in real-time.

III.3 Graphical User interface

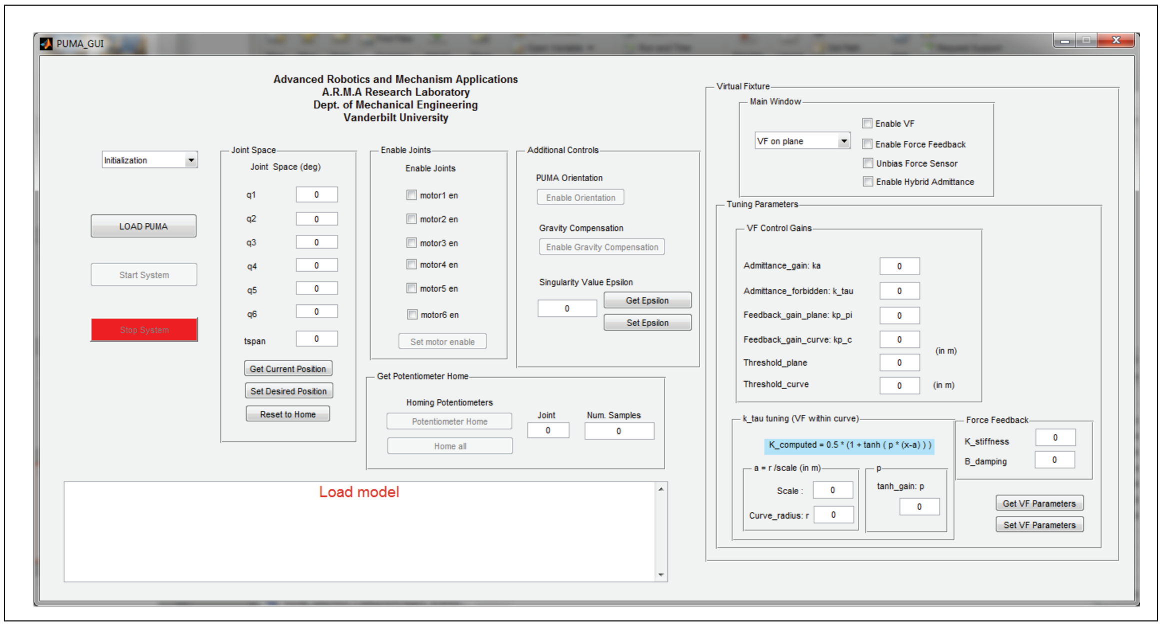


Figure III.9: User Control Interface

The user control interface was implemented to allow access to all the control modes of the PUMA and allow safe transition between all the modes. The interface allows us to switch between the different modes of operation of the PUMA and also allows us to tune the virtual fixture parameters of the controller in real time. Figure III.10 shows the different modes of operation of the PUMA controller. The *Task*

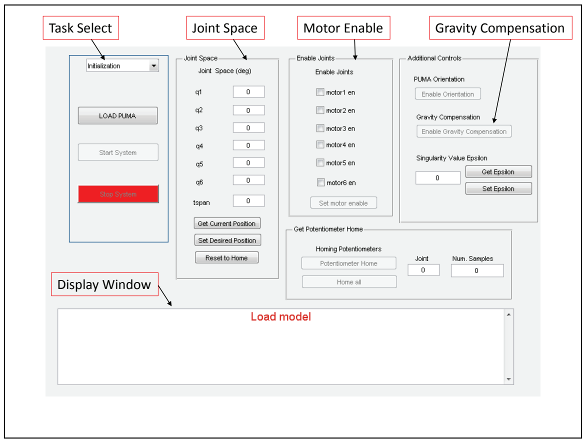
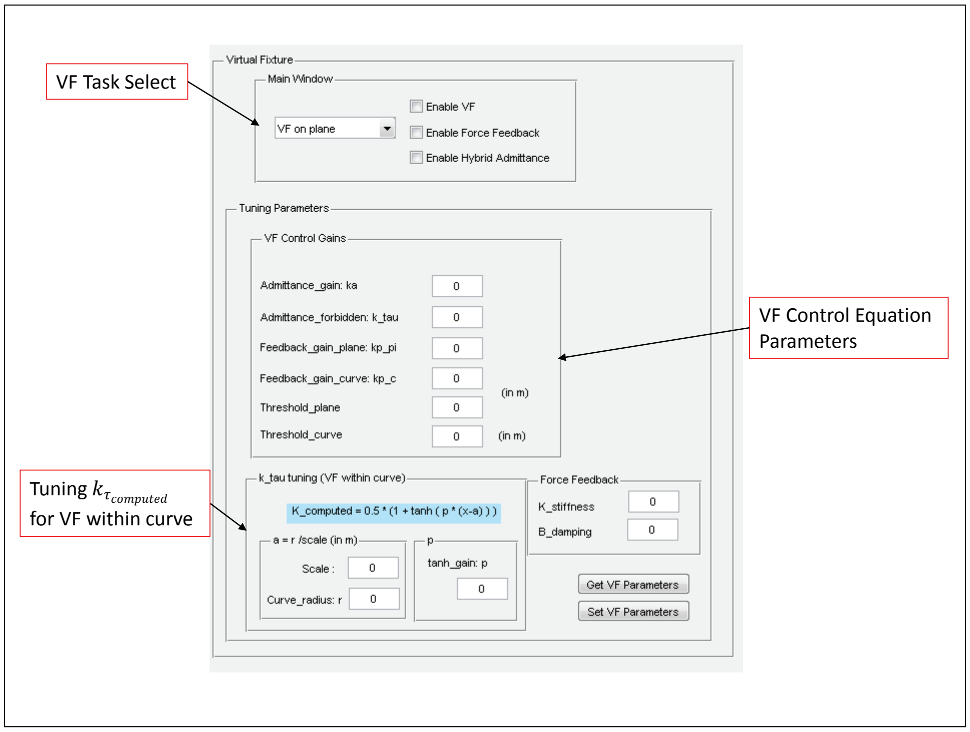


Figure III.10: Modes of Operation in GUI

Select panel is used to select the mode of operation of the PUMA. The *Load Model* button loads the model to the xPC target and then we can start the system. The pop-up menu allows to to select between the joint space and task space modes. The *Joint Space* panel is used to control the joint values when in the joint space mode. The *Motor Enable* panel allows us to disable individual motors. As a safety measure, whenever the motors are disabled, we switch to the joint space mode and update the desired joint values $\mathbf{q}_d$ before enabling the motors.

We can also switch to the gravity compensation mode using the GUI. The gravity compensation mode sets the proportional and derivative gains in the *PD + Inverse Dynamics* subsystem to zero. As soon as the gravity compensation mode is enabled, the trajectory planner is switched to the joint space mode. When we disable the gravity compensation mode, we wait till the joint space controller catches up with the current joint values and then enable the proportional and derivative gains.



VF on plane

Enable VF
Enable Force Feedback
Enable Hybrid Admittance

Admittance_gain: ka 0
Admittance_forbidden: k_tau 0
Feedback_gain_plane: kp_pi 0
Feedback_gain_curve: kp_c 0
Threshold_plane 0 (in m)
Threshold_curve 0 (in m)

k_tau tuning (VF within curve)
K_computed = 0.5 * (1 + tanh (p * (x-a)))
a = r /scale (in m)
Scale : 0
Curve_radius: r 0
p
tanh_gain: p 0

K_stiffness 0
B_damping 0

Figure III.11: Virtual Fixture Panel in GUI

The user interface also allows us to control the virtual fixture parameters (Figure III.11). The *Virtual Fixture* panel allows us to select between the three types of virtual fixture laws (follow plane, follow curve, stay within curve). The parameters used in virtual fixture laws can also be easily modified during operation. Virtual fixtures can only be enabled when we are in the task space mode and we can either telemanipulate the PUMA robot with a master device or cooperatively manipulate the robot using the force sensor.

CHAPTER IV

EXPERIMENTAL VALIDATION

IV.1 Virtual Fixtures on PUMA560

The proposed virtual fixtures defined in chapter II were successfully implemented on a PUMA560 robotic arm. The goal of implementing virtual fixtures on the PUMA was to verify the derived VF equations and also show the performance improvement in carrying out specified tasks.

Prior to commencing with experiments, the admittance gains for the virtual fixture laws were tuned to provide seamless and natural telemanipulation behavior. The values of all the impedance gains used in the experiments are reported in table IV.1.

Table IV.1: Admittance gains used for virtual fixture evaluation

VF Type	K_a	K_τ	K_{p_π}	K_{p_c}	VF Equation
On Plane	1	0.1	0.01	–	II.30
On Curve	1	0.1	0.01	0.01	II.32
Within Curve	1	–	0.01	0.01	II.33

The Virtual fixture equation for the within curve case has a K_τ value that is computed from Equation II.34. The proportional gain p in Equation II.34 is chosen to be 100 and the offset from origin a is chosen to be $1/10^{\text{th}}$ the radius of the curve.

Six users were asked to perform the experiments. Each of the users had to perform three tasks:

1. Follow a Plane
2. Follow a Curve
3. Stay within a curve

The users had to perform each task 10 times, five times with no assistance provided and the other five with virtual fixtures. This gave us a total of 60 data sets for each task and 180 data sets overall. In the experiments conducted, the users had to telemanipulate the PUMA arm using a haptic master device (Omega 7). The users were given enough time to get accustomed to telemanipulating the PUMA arm using the master device. They were then asked to perform each task five times without assistance and then five times with assistance. A passive optical marker was mounted to the tool attached to the PUMA end effector and a 3D-Optical Tracker(NDI Vicra) was used to track the movement of the marker. The tracker has an accuracy of 0.2mm in working volume. In addition to collecting the data from the tracker, every fifth trial (with and without VF) was recorded on video as well.

A statistical test was used to compare the performance between the trials without VF assistance and trials with VF assistance. Whenever two populations are compared, either the z-test or the t-test can be used. The choice of the test depends on the number of samples and whether the samples are independent or dependent.

Independent samples means that the two populations comprise of different people e.g. testing the effect of an anti-depressant on two groups of users where one group is given the medicine and the other a placebo. In this case, two completely different sets of people are being tested and therefore the samples are considered independent. With independent samples, when the population sizes (n_1, n_2) are greater than 30,

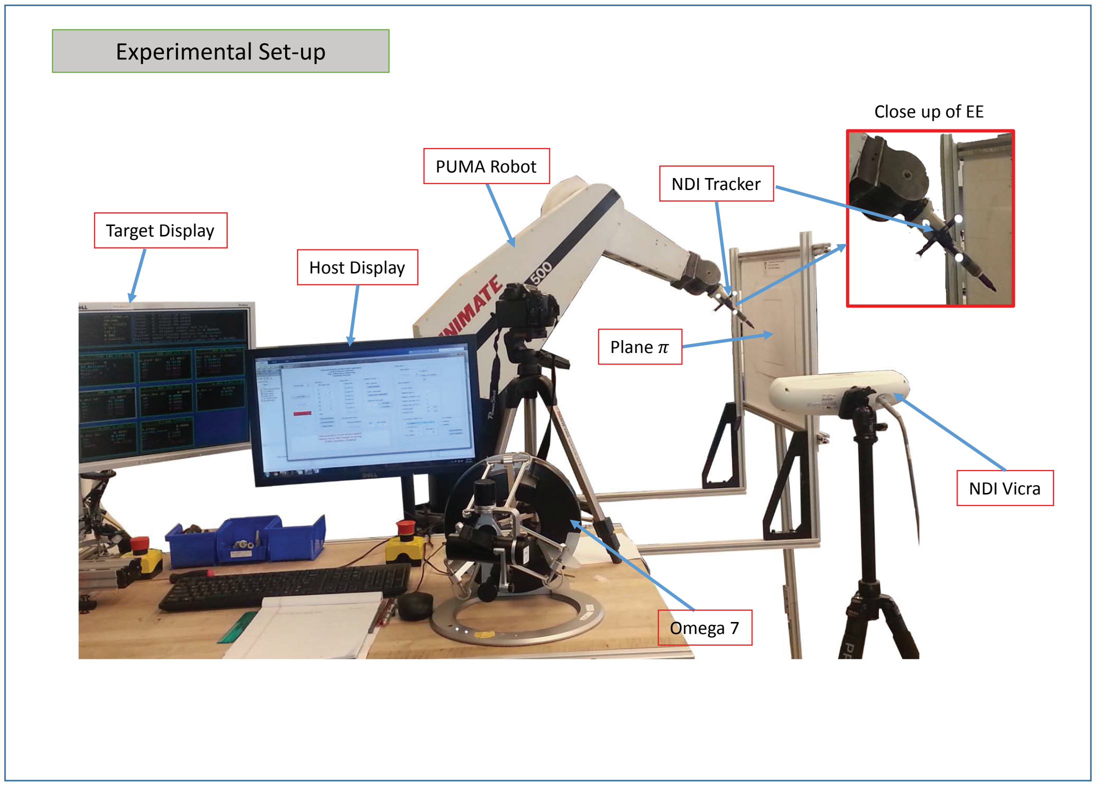


Figure IV.1: Experimental setup for virtual fixture evaluation

the z-test is used; otherwise, if the sample size is small i.e. $(n_1, n_2) < 30$, the p-test is used [27].

Dependent samples means that corresponding values in the two populations are paired/dependent e.g. testing the performance of users in tasks with VF assistance and without VF assistance. With dependent samples, the paired t-test is used irrespective of the sample size. The null hypothesis being tested would be that the trials with and without virtual fixtures belong to the same distribution group.

Virtual Fixture constraining movement to a plane

Methodology

The first task was to follow a plane which is fixed with respect to the PUMA robot base. The metric used for the paired t-test was the mean deviations from the plane. For each user there are two vectors $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ containing five corresponding values of mean deviations from the plane where,

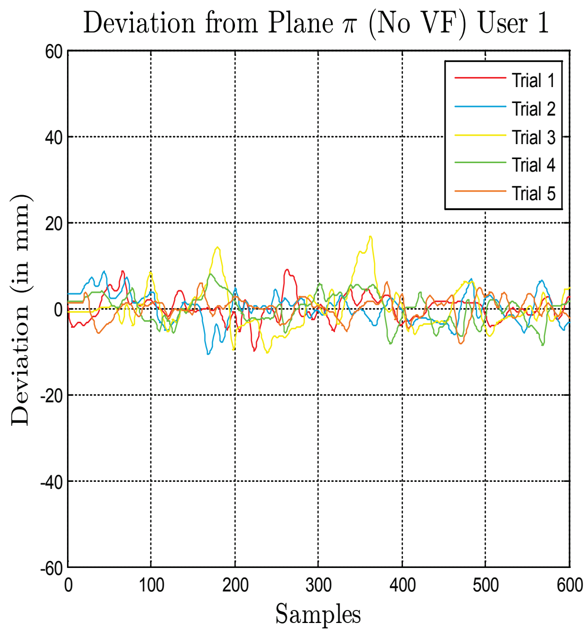
$$\overline{\mathbf{x}} : \text{mean deviation from the plane without VF}$$

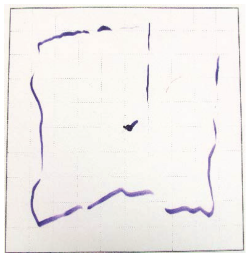
$$\overline{\mathbf{y}} : \text{mean deviation from the plane with VF}$$

The paired t-test tests the null hypothesis that the data in the vectors $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ are independent random samples from normal distributions with equal means and equal but unknown variances, against the alternative that the means are not equal. However, the data-sets are not expected to have equal variances. Therefore, the paired t-test is used without assuming equal variances. The right-tailed test is also performed along with the paired t-test which tests the alternative hypothesis whether the mean of $\overline{\mathbf{x}}$ is greater than the mean of $\overline{\mathbf{y}}$. Therefore, if the null hypothesis is rejected, we can say that the case with VF is not only different from the case without VF but also yields a better result. A p-value < 0.05 would indicate a rejection of the null hypothesis at the 5% significance level.

Figures IV.2 (a)-(e) present the analysed data collected during the experiments. The figures show the deviation of the robot end effector and the RMS tracking error from the plane $\boldsymbol{\pi}$ with and without virtual fixtures, for each user. The deviations and errors reduce by a large margin with virtual fixture assistance.

(a)User 1 (No VF)





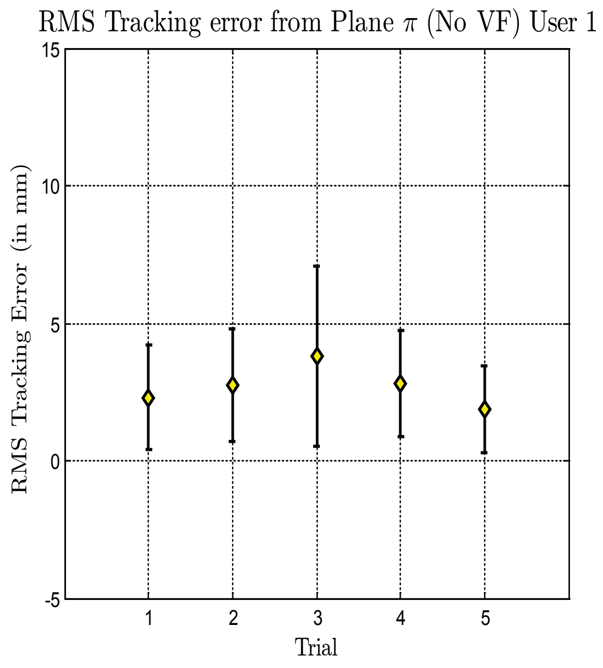
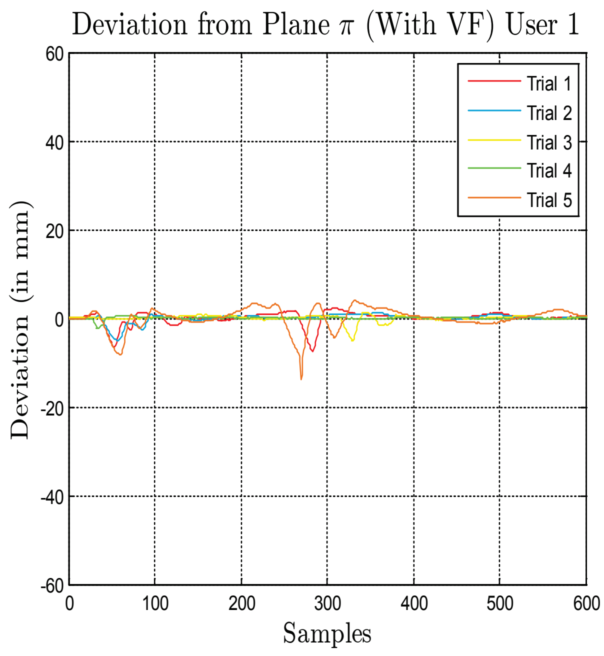
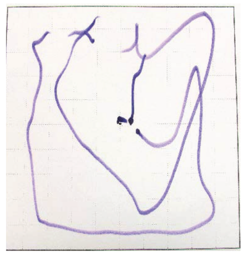


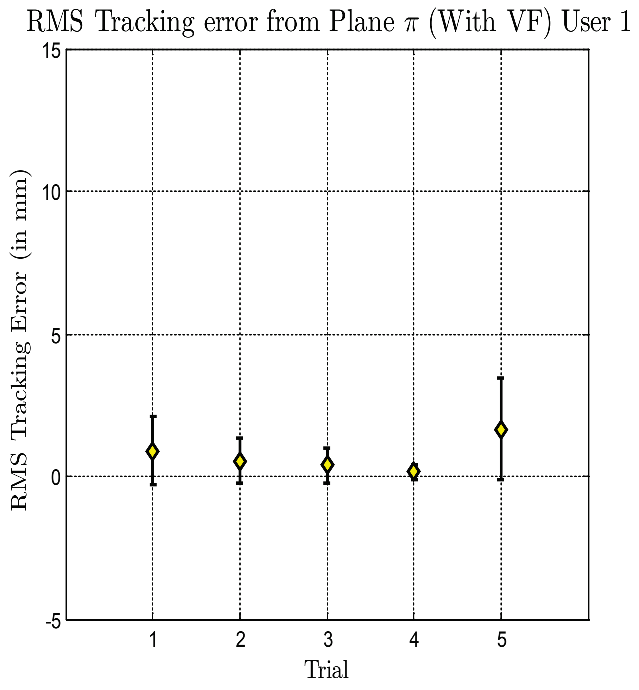
Figure IV.2: Virtual Fixture Task, Follow Plane

(a)User 1 (With VF)



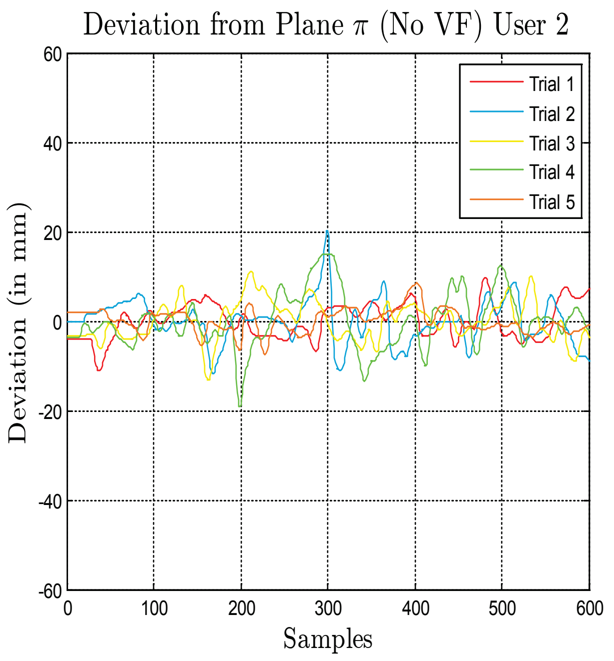


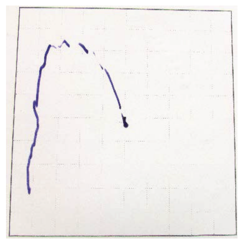
Follow Plane: With VF



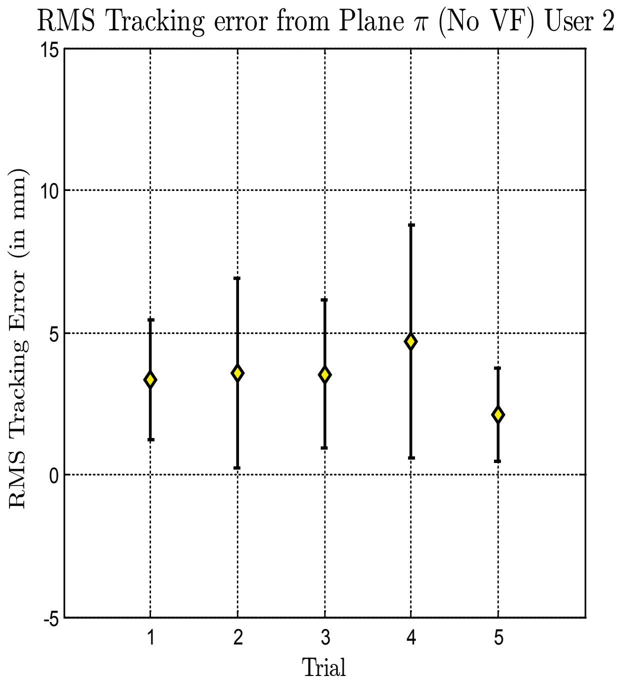
Virtual Fixture Task, Follow Plane

(b)User 2 (No VF)



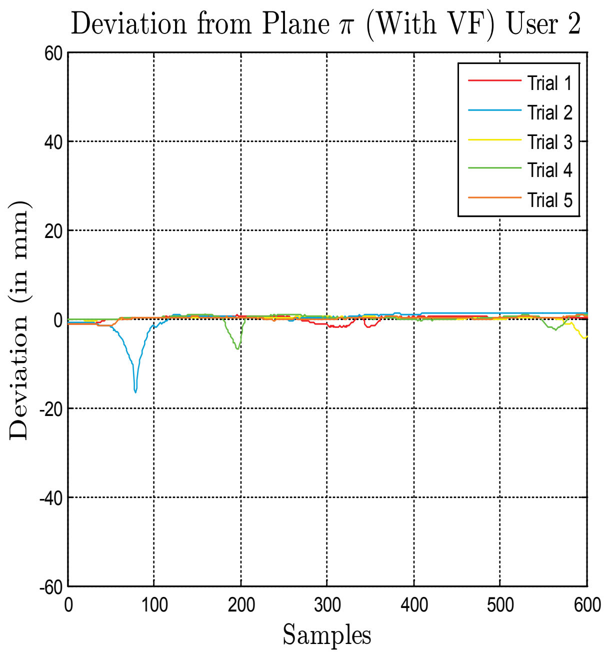


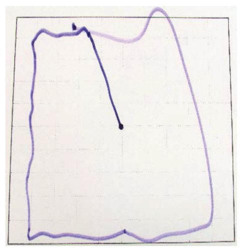
Follow Plane: No VF



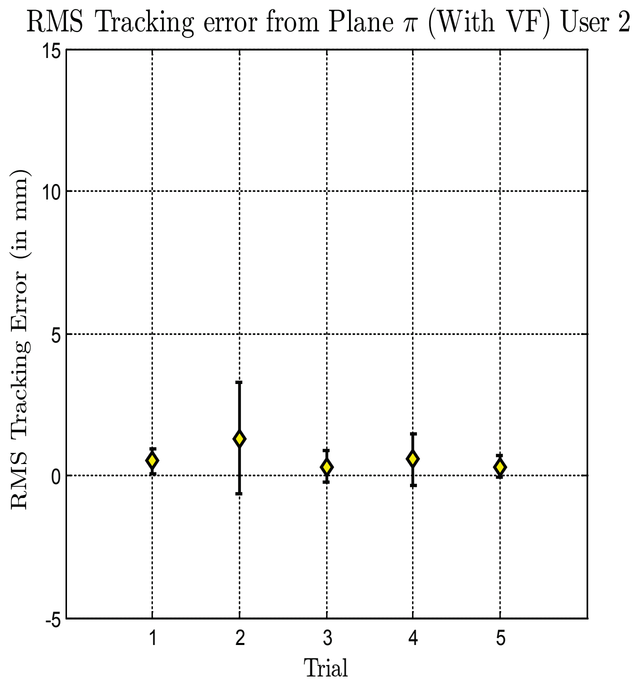
Virtual Fixture Task, Follow Plane

(b)User 2 (With VF)



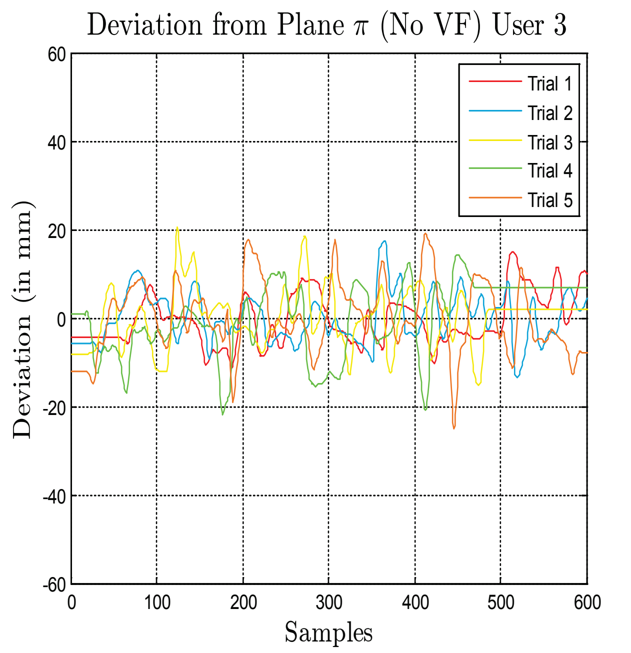


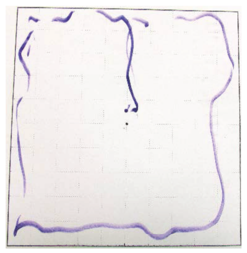
Follow Plane: With VF



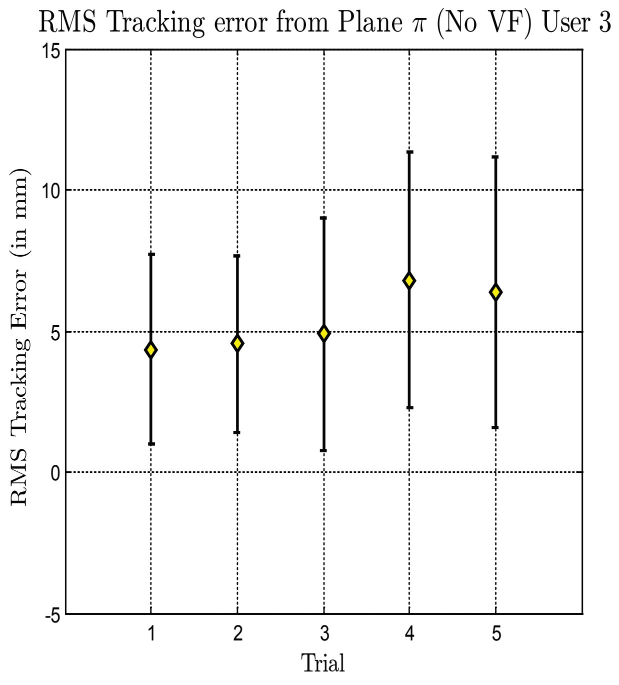
Virtual Fixture Task, Follow Plane

(c)User 3 (No VF)





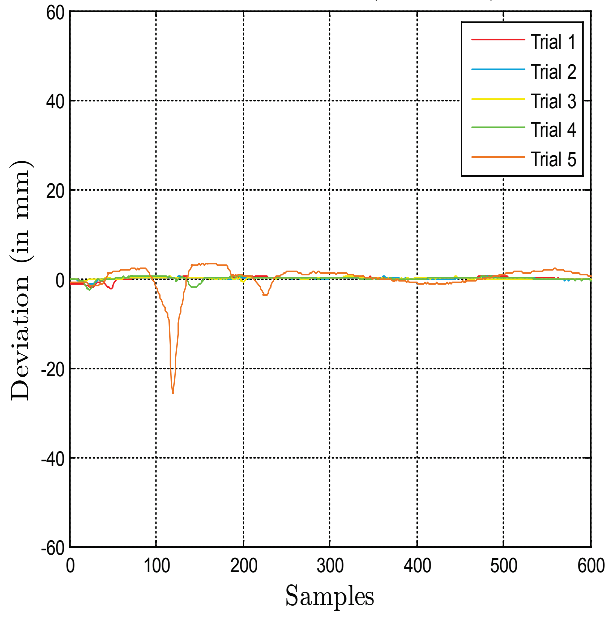
Follow Plane: No VF

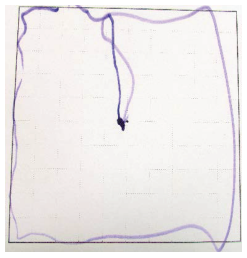


Virtual Fixture Task, Follow Plane

(c)User 3 (With VF)

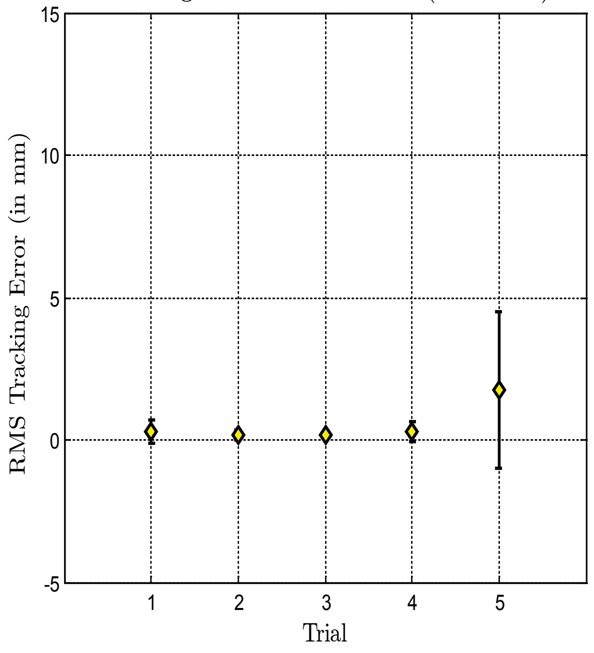
Deviation from Plane π (With VF) User 3





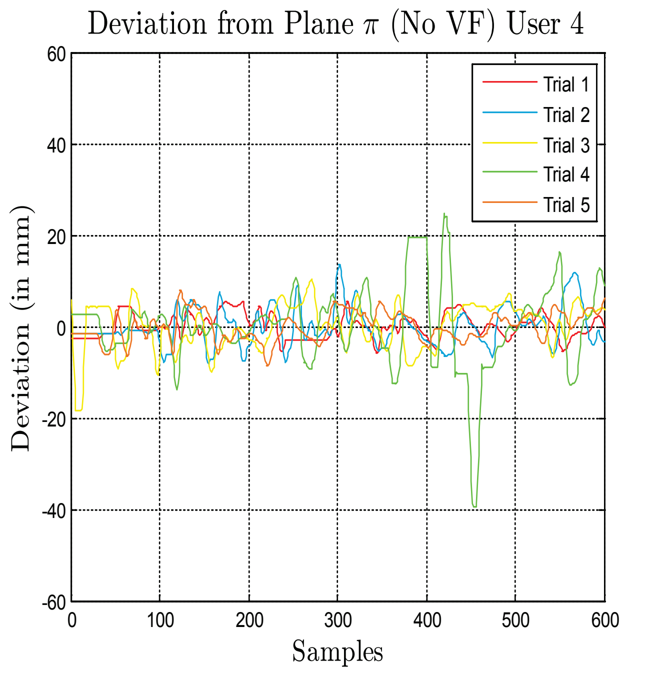
Follow Plane: With VF

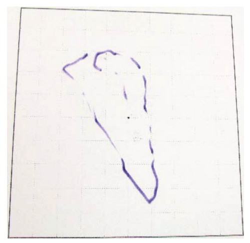
RMS Tracking error from Plane π (With VF) User 3



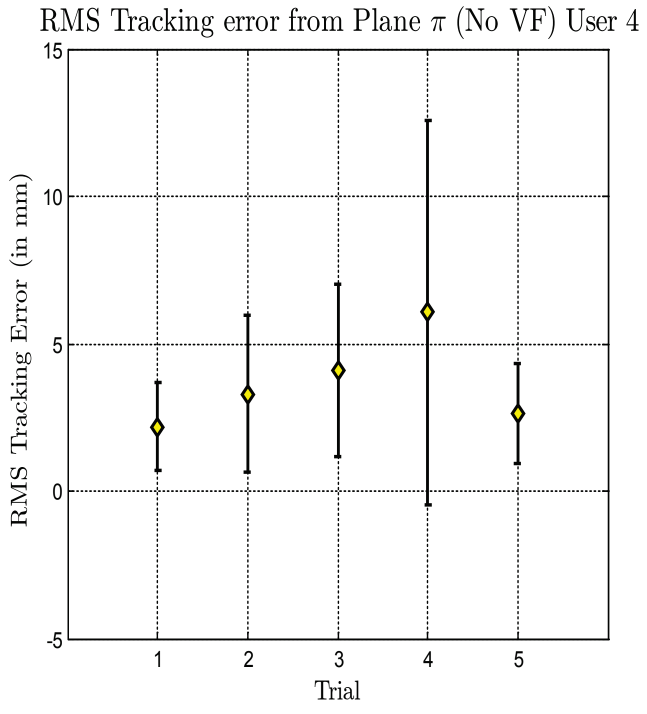
Virtual Fixture Task, Follow Plane

(d)User 4 (No VF)



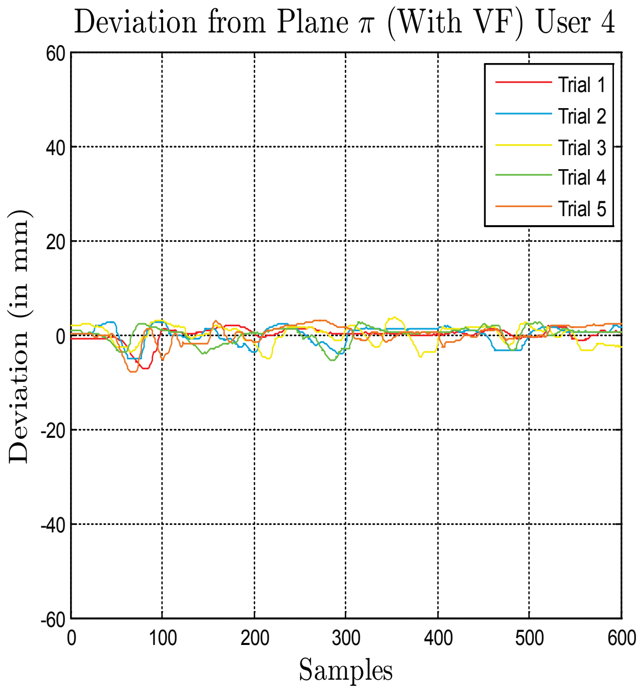


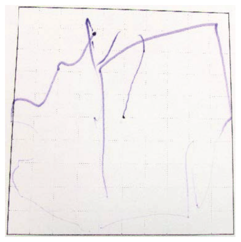
Follow Plane: No VF



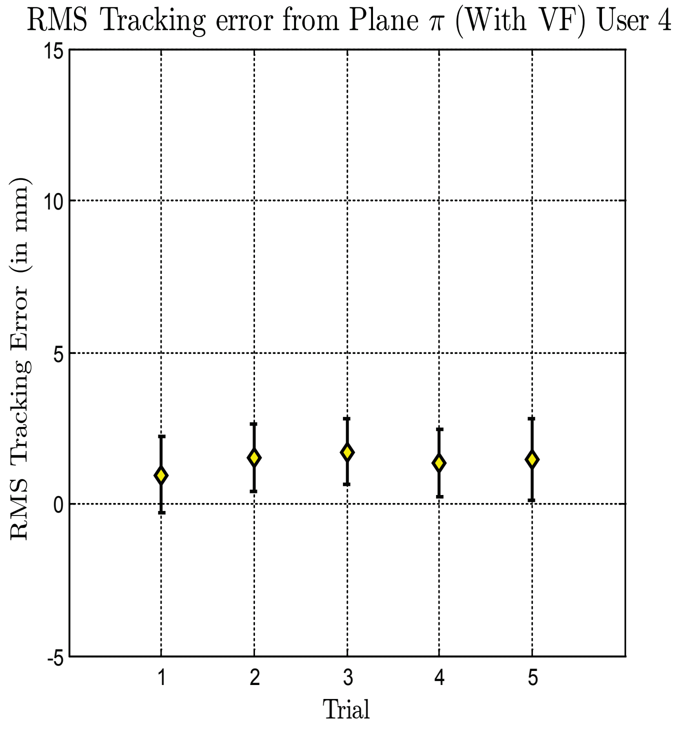
Virtual Fixture Task, Follow Plane

(d)User 4 (With VF)



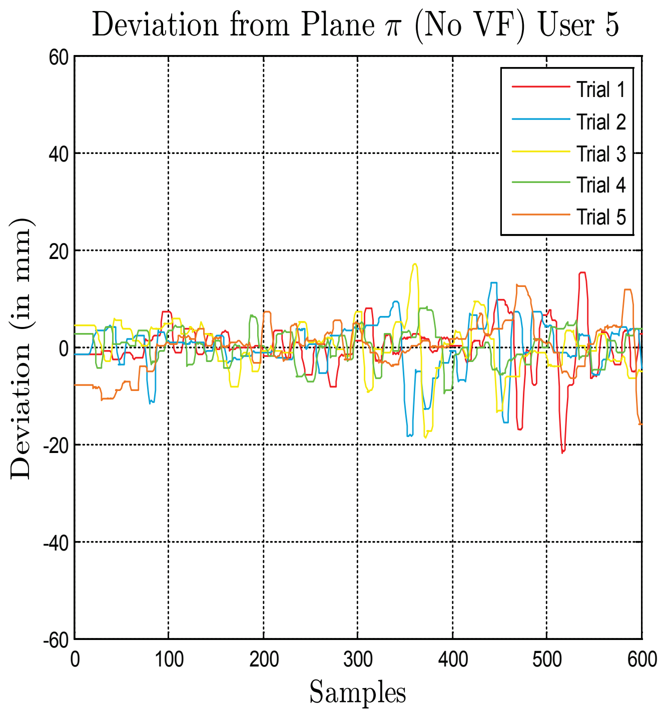


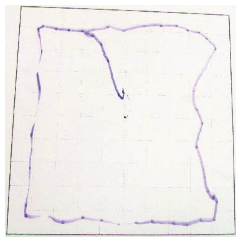
Follow Plane: With VF

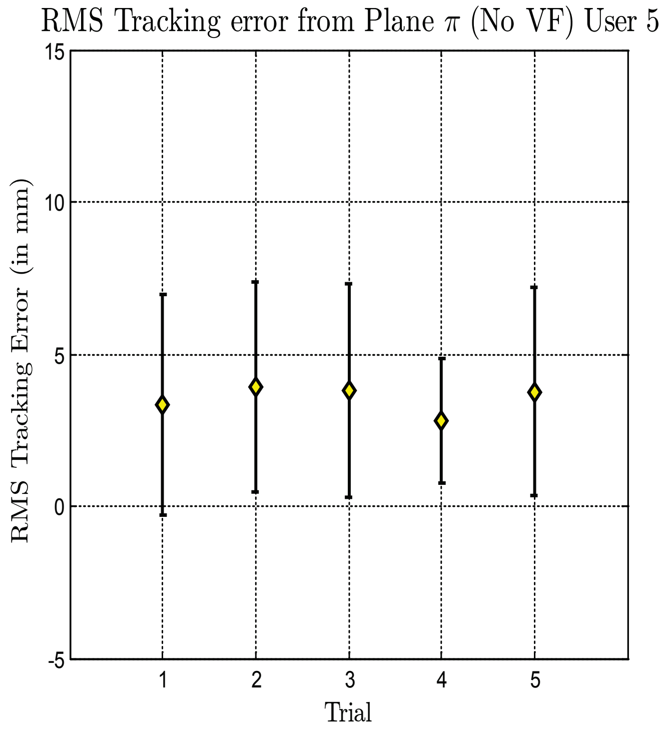


Virtual Fixture Task, Follow Plane

(e)User 5 (No VF)

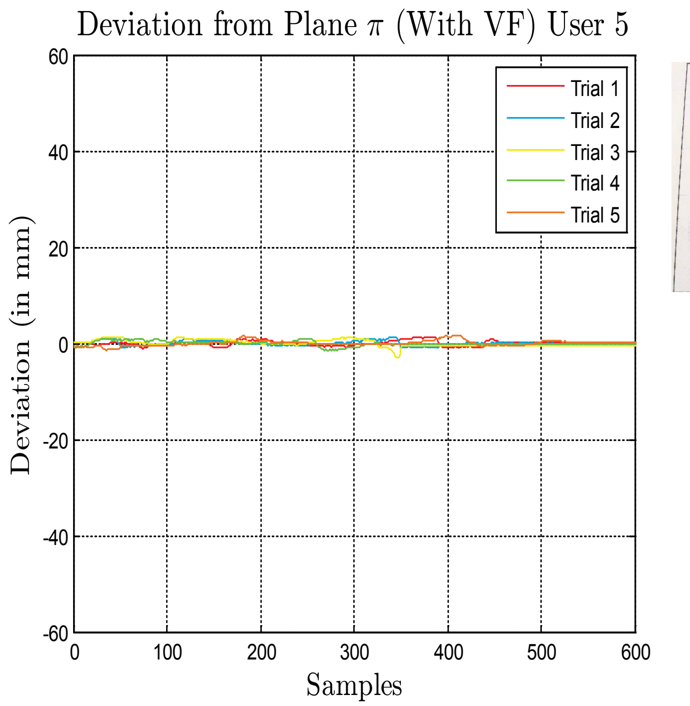


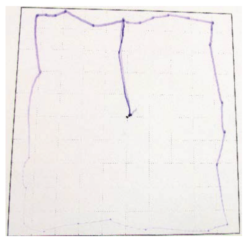


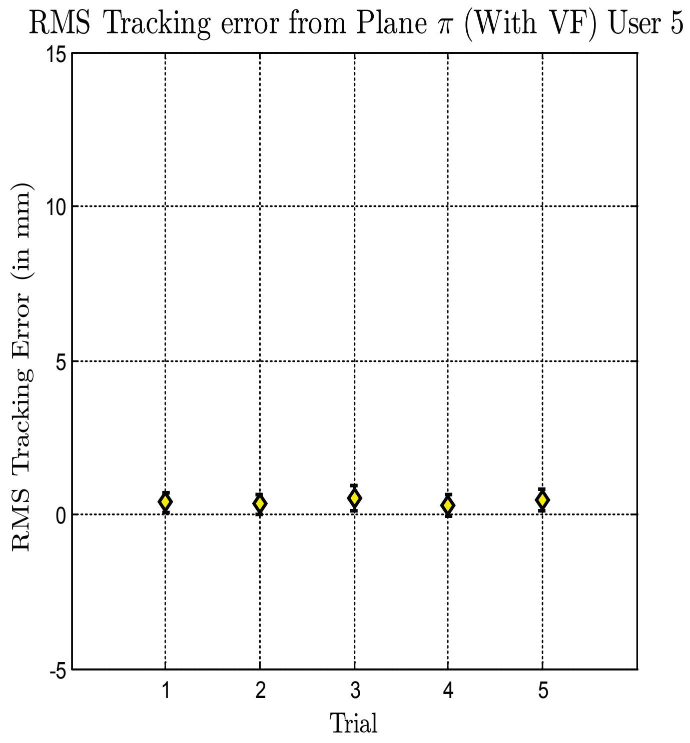


Virtual Fixture Task, Follow Plane

(e)User 5 (With VF)

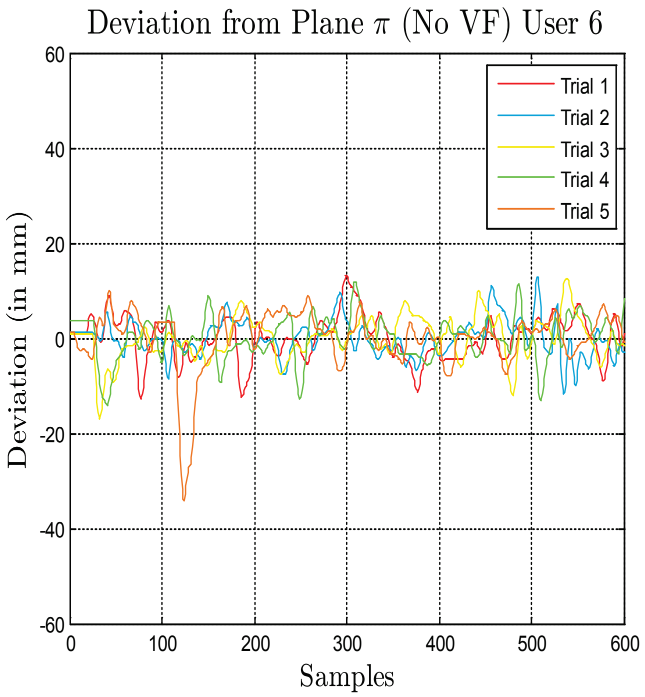


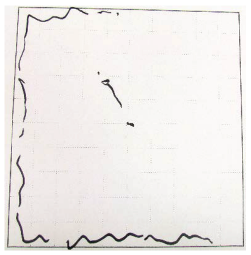




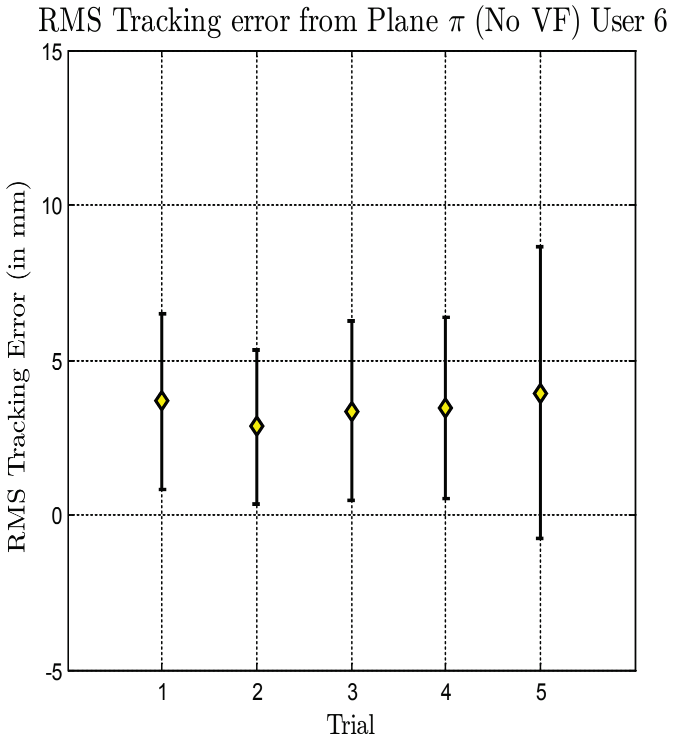
Virtual Fixture Task, Follow Plane

(f)User 6 (No VF)



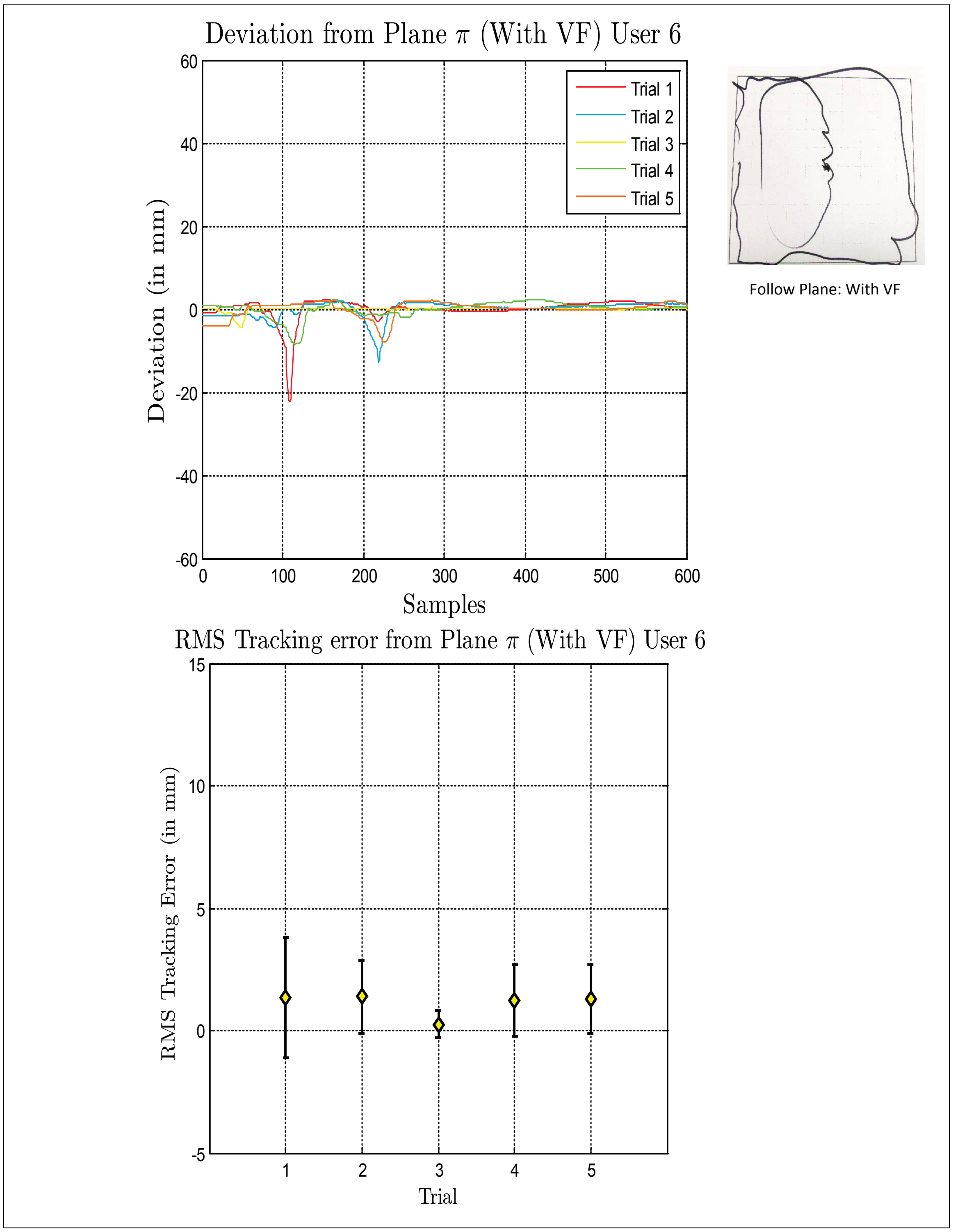


Follow Plane: No VF



Virtual Fixture Task, Follow Plane

(f)User 6 (With VF)



Virtual Fixture Task, Follow Plane

Paired t-test Results

	Without VF	With VF	Paired t-test	
User	$\bar{x}$: *mean deviations*	$\bar{y}$: *mean deviations*	p-value	Null-Hypothesis
User 1	2.286286	0.899639	0.000743 < 0.05	Rejected
	2.770792	0.541385		
	3.777216	0.38799		
	2.811424	0.161702		
	1.87956	1.650919		
User 2	3.321986	0.520559	0.000482 < 0.05	Rejected
	3.570267	1.305667		
	3.526338	0.316155		
	4.689462	0.561243		
	2.120143	0.299053		
User 3	4.344325	0.297085	0.000051 < 0.05	Rejected
	4.558456	0.178431		
	4.887396	0.167147		
	6.803751	0.288162		
	6.394346	1.734519		

Table IV.2: Paired t-test Results for Follow Plane Task

Paired t-test Results

	Without VF	With VF	Paired t-test	
User	$\bar{x}$: *mean deviations*	$\bar{y}$: *mean deviations*	p-value	Null-Hypothesis
User 4	2.188765	0.947103	0.014044< 0.05	Rejected
	3.301404	1.524002		
	4.102778	1.723187		
	6.053291	1.339793		
	2.638096	1.460541		
User 5	3.320317	0.404319	0.000031< 0.05	Rejected
	3.911539	0.332163		
	3.803184	0.533755		
	2.820691	0.299538		
	3.747581	0.449781		
User 6	3.658596	1.353678	0.000018< 0.05	Rejected
	2.837999	1.387769		
	3.352513	0.253855		
	3.458562	1.215268		
	3.929255	1.293254		

Paired t-test Results for Follow Plane Task

Experimental Results

Figures IV.2 (a)-(e) show that the deviations from the plane is much less when the user has virtual fixture assistance and the user is able to follow the plane with much greater accuracy. In the paired t-test results (Table IV.2), the null hypothesis is rejected in each case. Therefore, there are two conclusions that can be drawn:

1. The strong rejection of the null hypothesis at $p = 0.05$ indicates that the two cases (with and without VF) are different.

2. The null hypothesis was rejected in the right tailed paired t-test indicating that the alternative hypothesis is accepted. In our case the alternative hypothesis is that the mean deviations in the case with no VF is greater than the case with VF. This means that the case with VF is able to follow the plane with much greater accuracy.

Virtual Fixture constraining movement along a curve

Methodology

The second task was to follow a curve on a plane which is fixed with respect to the PUMA robot base. We measure the accuracy of tracking the curve and the plane. Therefore, the two metrics considered are:

1. Mean deviation from the curve.
2. Mean deviation from the plane.

The combined metric considered for the paired t-test is:

$$\begin{aligned}\text{Weighed Deviation } &= \text{mean deviation from the curve } + \\ &\quad (0.75 * \text{mean deviation from the plane})\end{aligned}$$

For each user there are two vectors $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ containing five corresponding values of weighed deviations from the plane where,

$$\begin{aligned}\overline{\mathbf{x}} &: \text{weighed deviation from the plane without VF} \\ \overline{\mathbf{y}} &: \text{weighed deviation from the plane with VF}\end{aligned}$$

As before, the test is conducted without assuming equal variances and with the right tailed test. Therefore, the rejection of the null hypothesis would indicate a performance improvement with VF.

Figures IV.3 (a)-(e) present the results collected during the experiments for this task. The figures show the deviation of the robot end effector and the RMS tracking errors from the plane $\boldsymbol{\pi}$ and the curve $\mathbf{c}$ with and without virtual fixtures, for each user.

Experimental Results

Figures IV.3 (a)-(e) show that the users deviated significantly less from the curve when virtual fixture assistance was provided. In the paired t-test results (Table IV.3), the null hypothesis is rejected in each case. As before, there are two conclusions that can be drawn from the results:

1. The strong rejection at $p = 0.05$ of the null hypothesis tells us the that the two cases (with and without VF) are different.

2. The null hypothesis was rejected in the right tailed paired t-test indicating that the alternative hypothesis is accepted. In our case the alternative hypothesis is that the mean deviations in the case with no VF is greater than the case with VF. This means that the case with VF is able to follow the curve with much greater accuracy.

(a)User 1 (No VF)

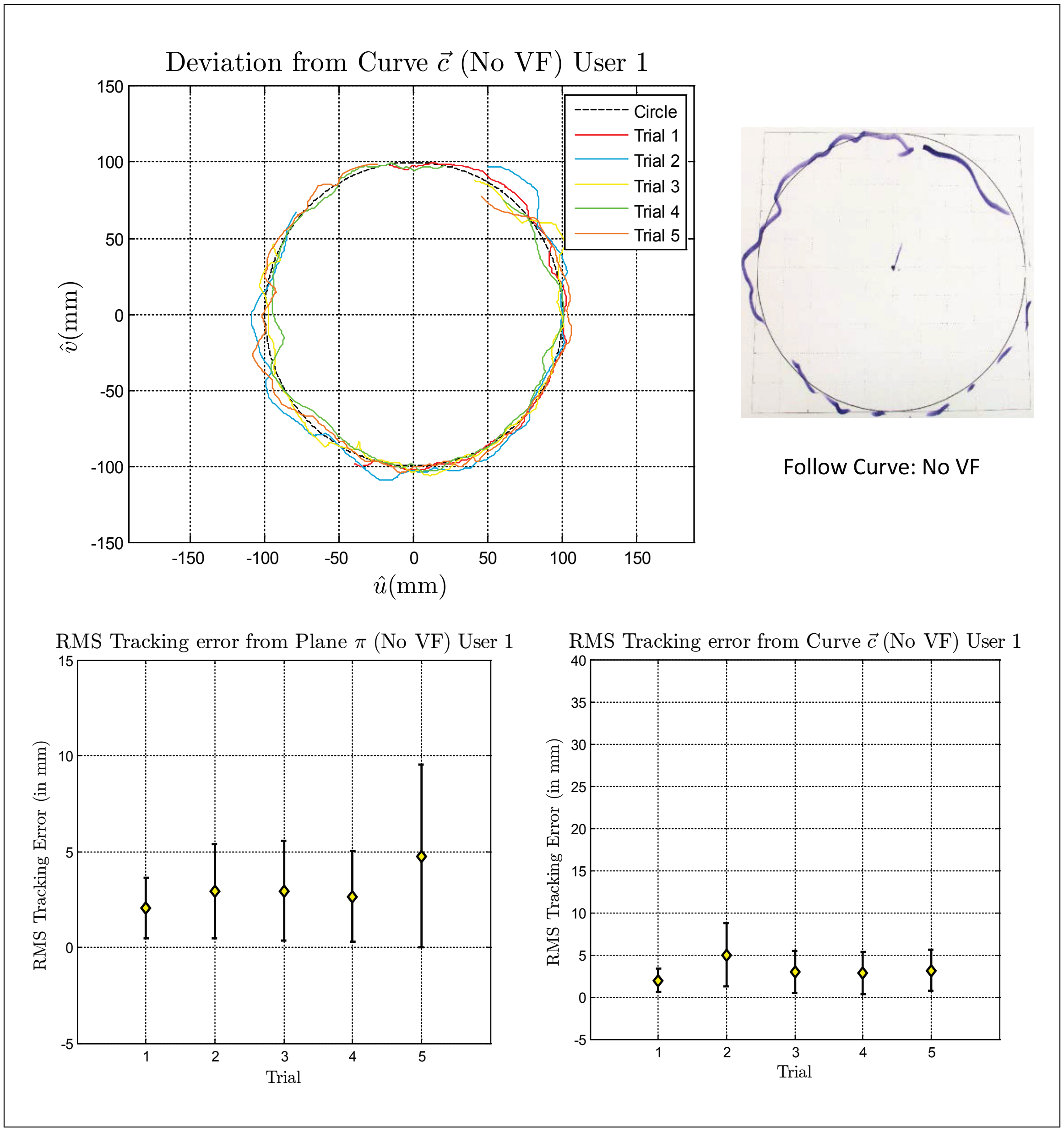
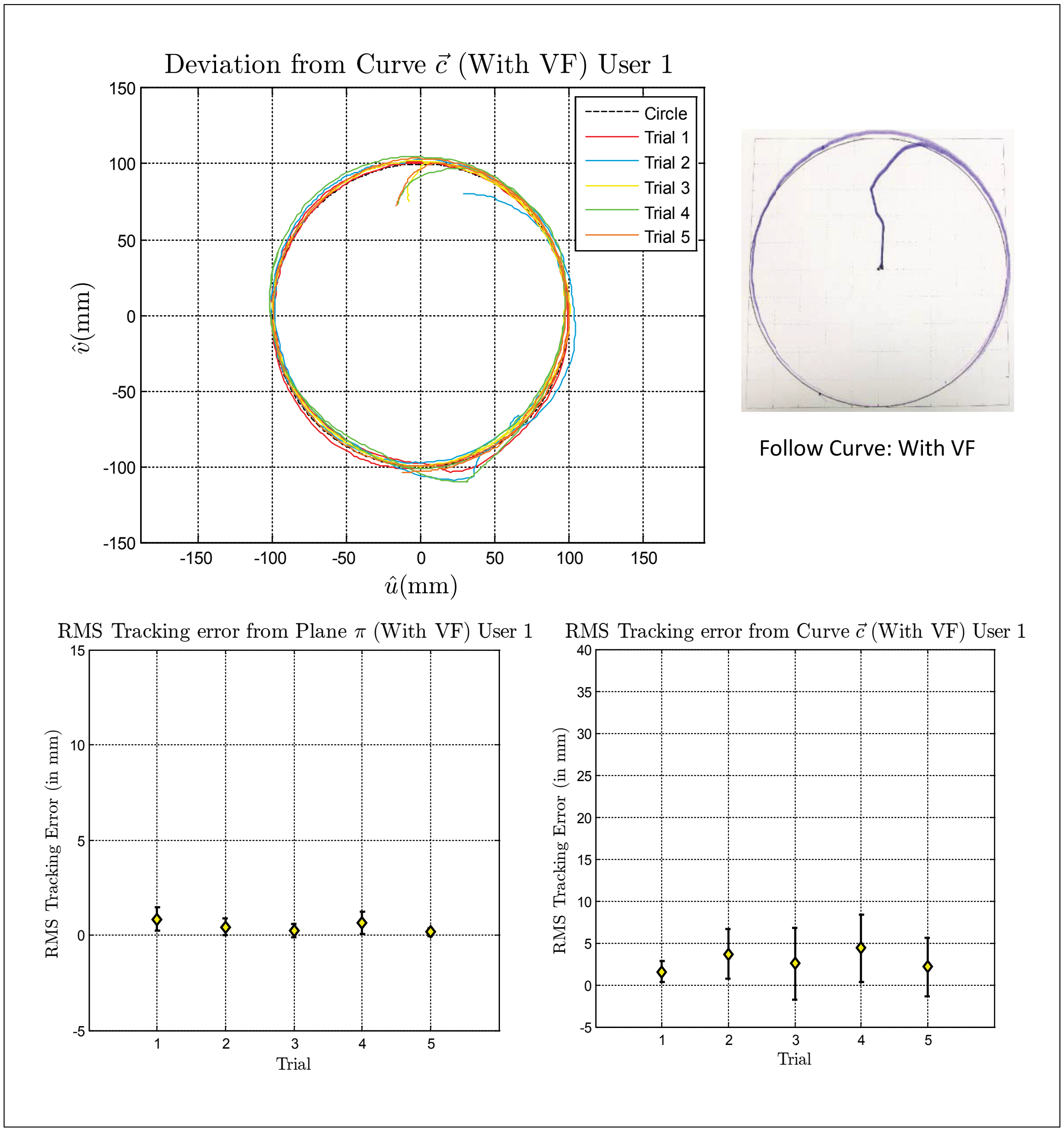


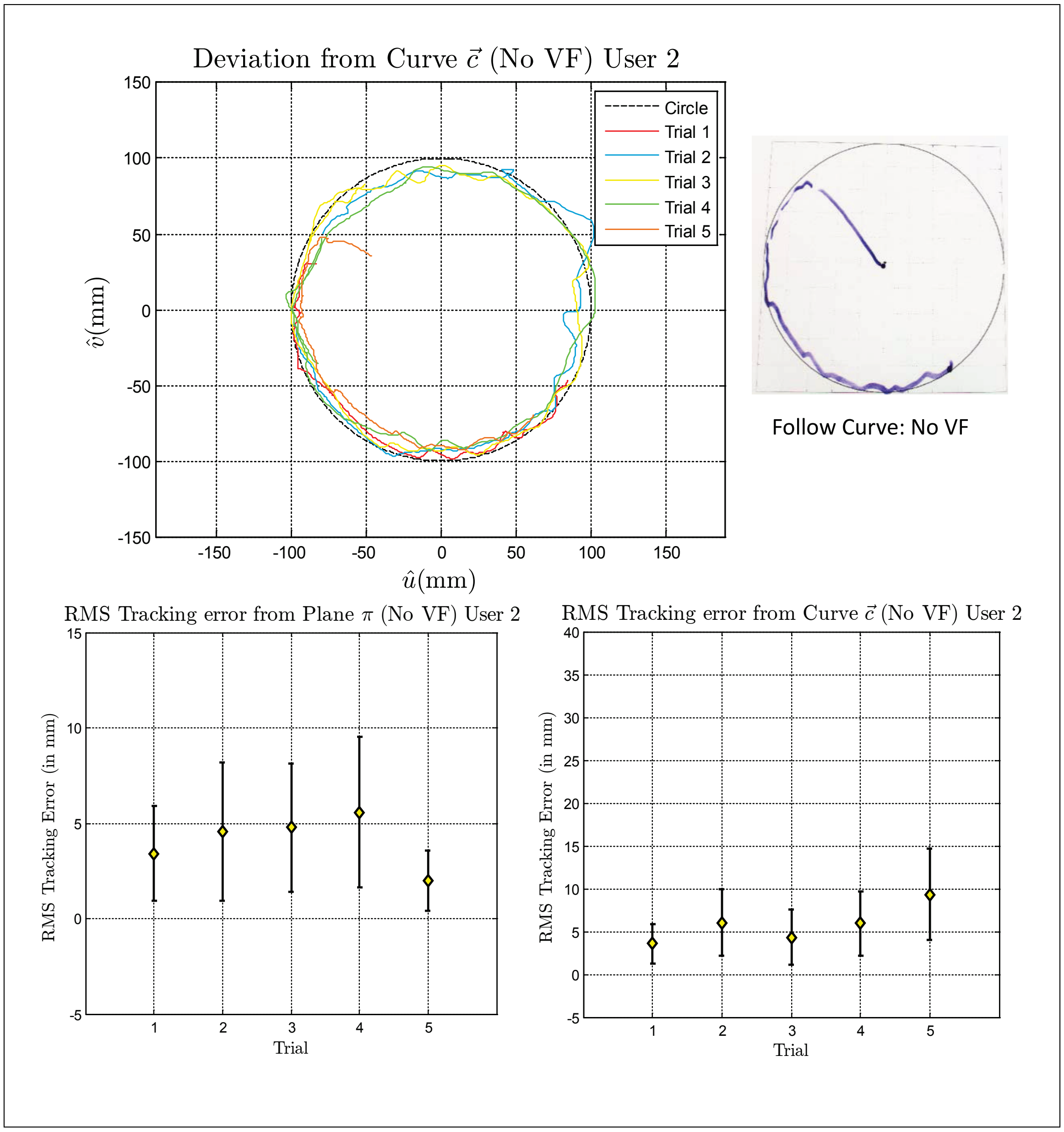
Figure IV.3: Virtual Fixture Task, Follow Curve

(a)User 1 (With VF)



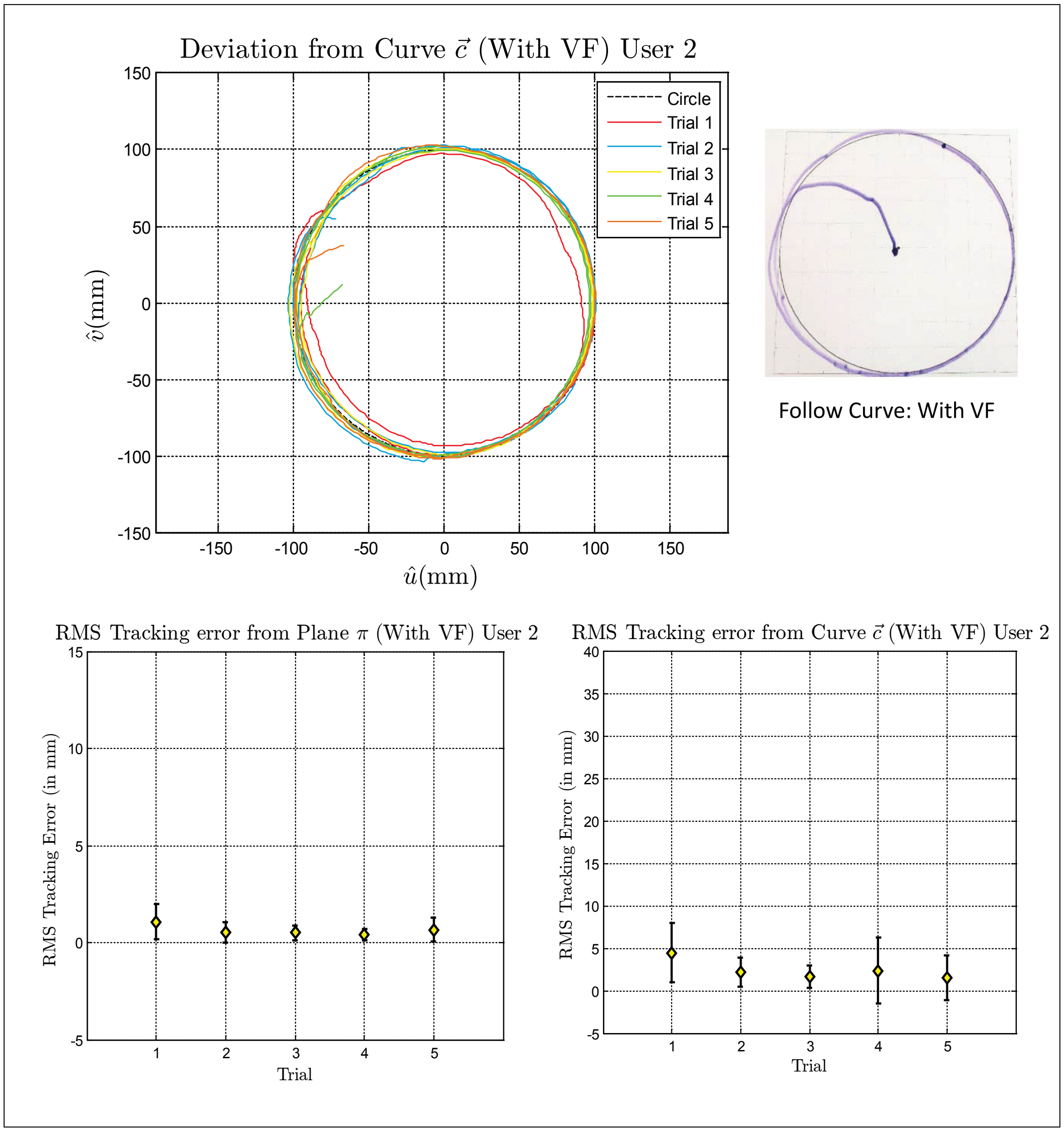
Virtual Fixture Task, Follow Curve

(b)User 2 (No VF)



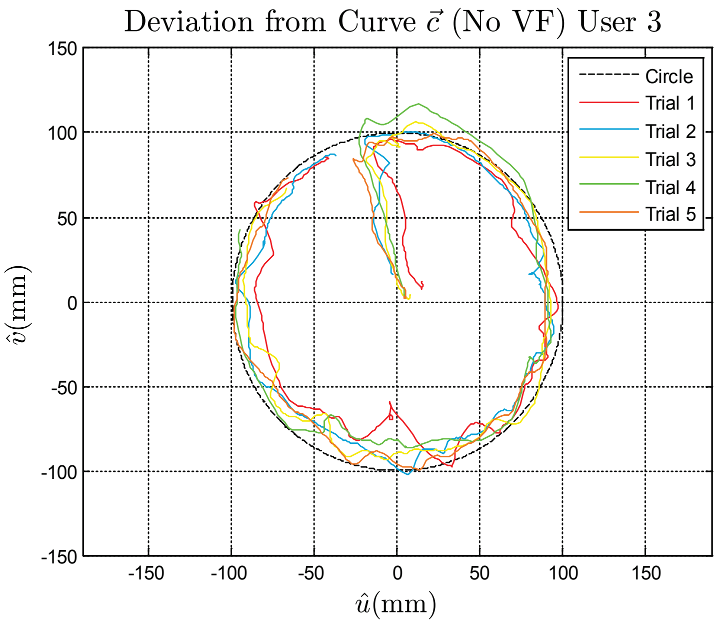
Virtual Fixture Task, Follow Curve

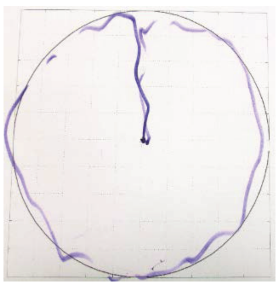
(b)User 2 (With VF)



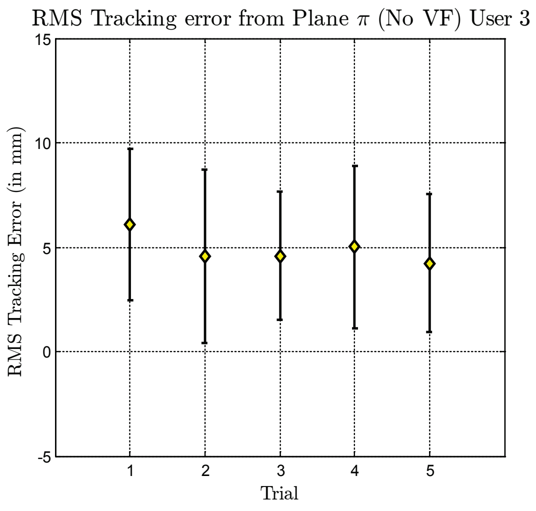
Virtual Fixture Task, Follow Curve

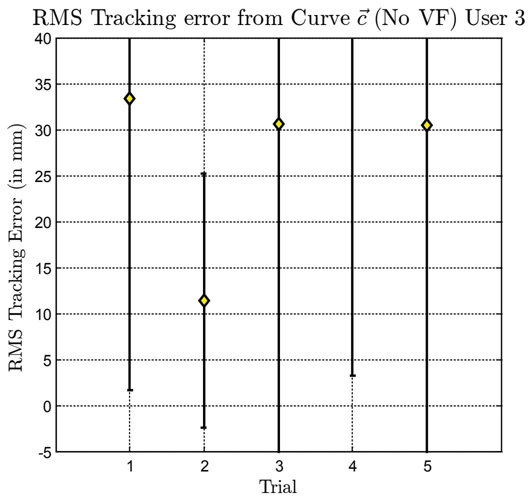
(c)User 3 (No VF)





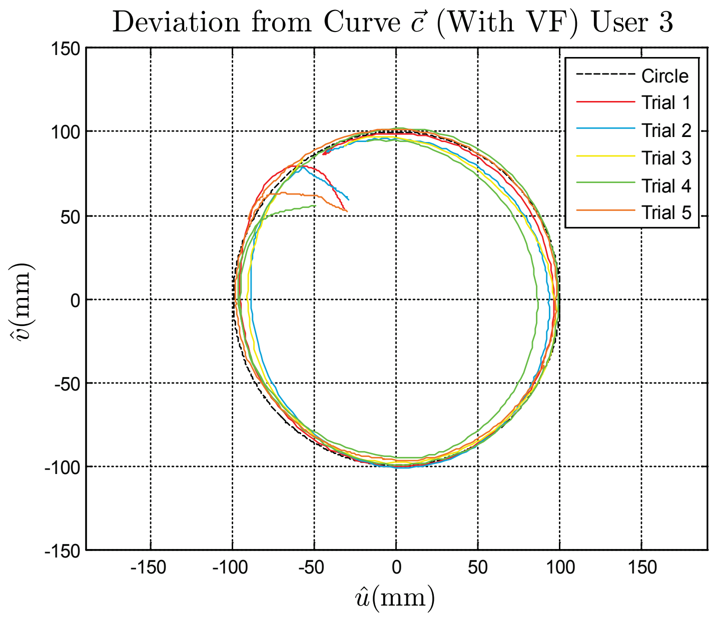
Follow Curve: No VF

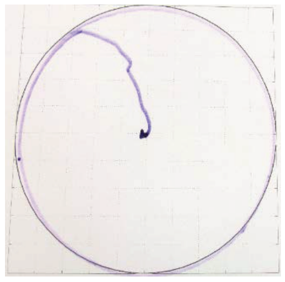




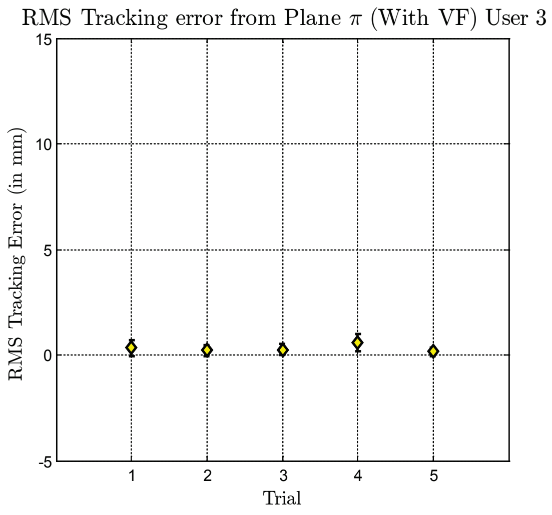
Virtual Fixture Task, Follow Curve

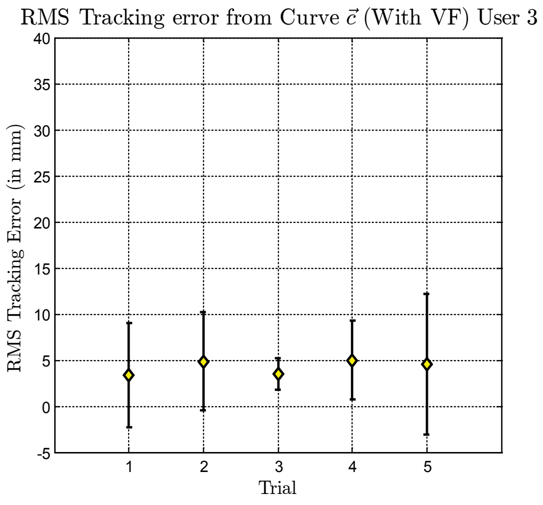
(c)User 3 (With VF)





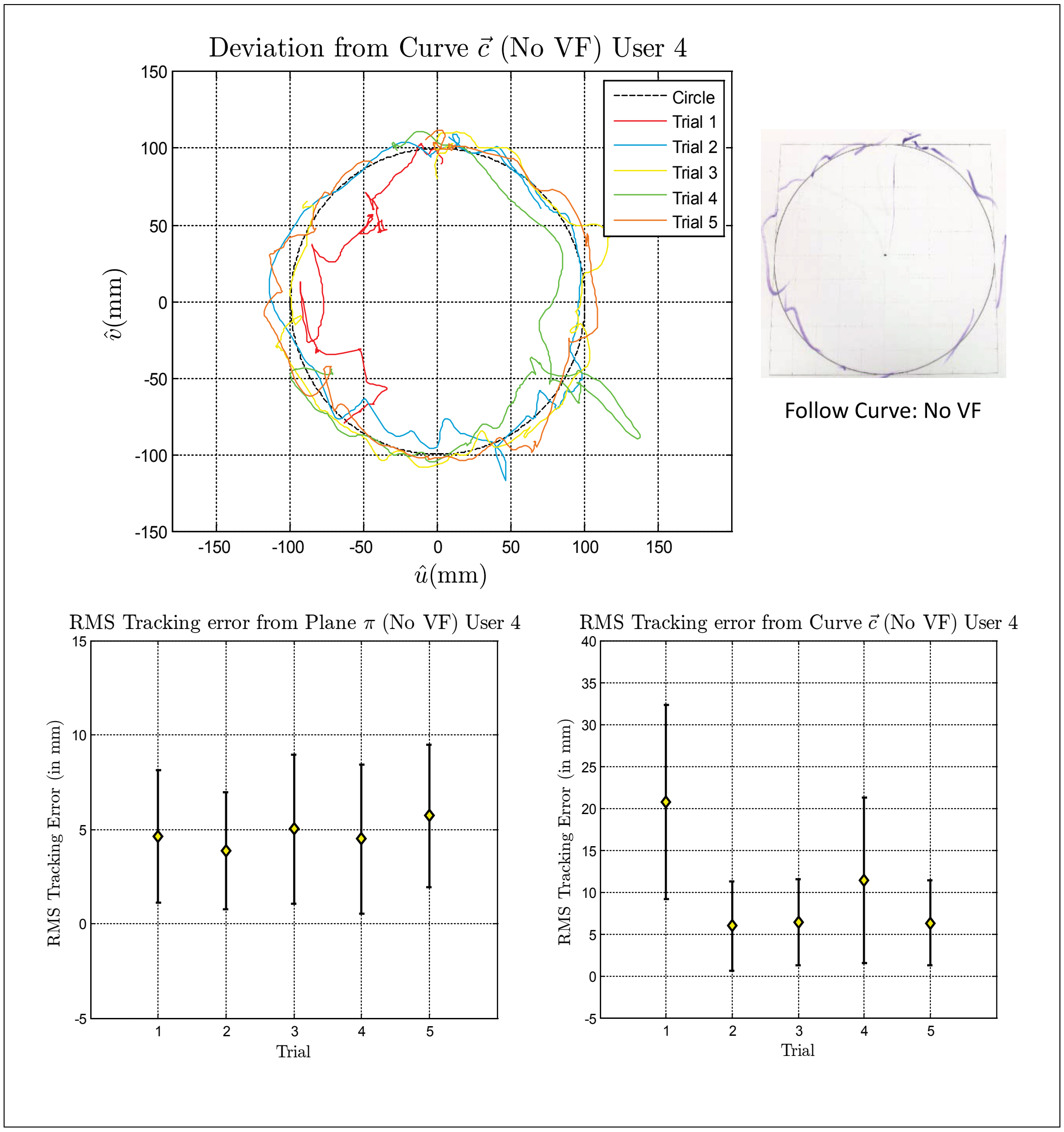
Follow Curve: With VF





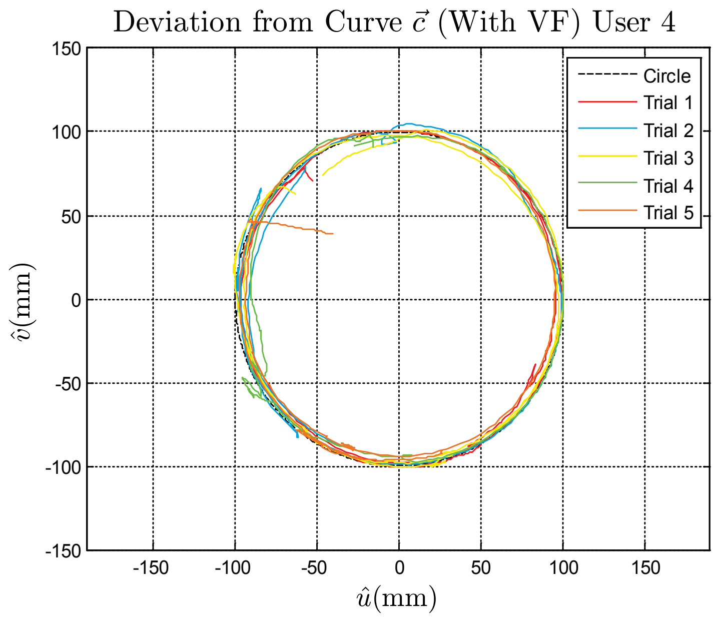
Virtual Fixture Task, Follow Curve

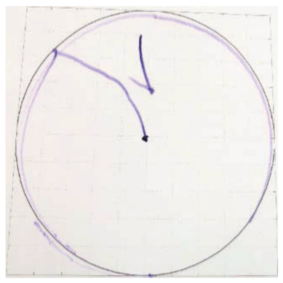
(d)User 4 (No VF)



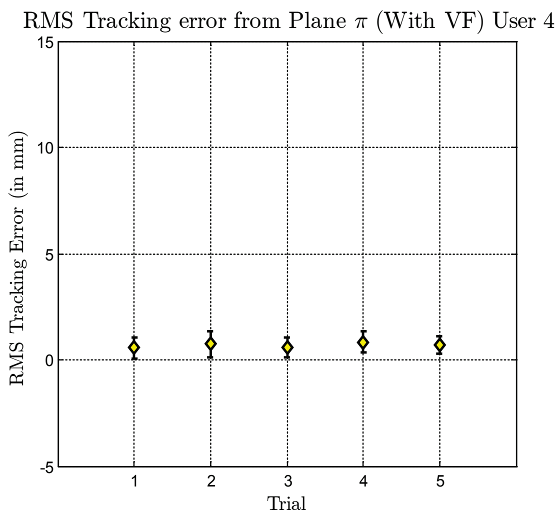
Virtual Fixture Task, Follow Curve

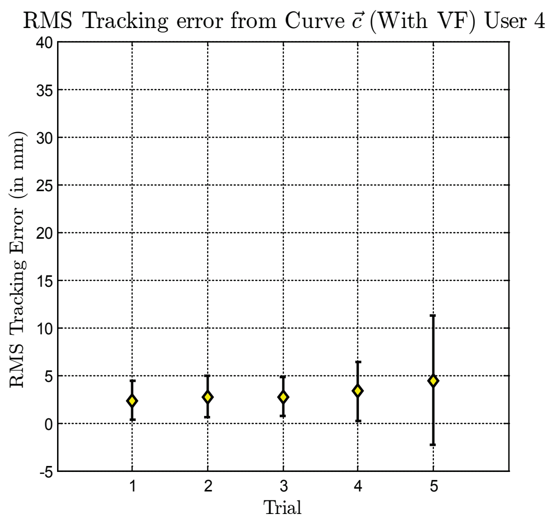
(d)User 4 (With VF)





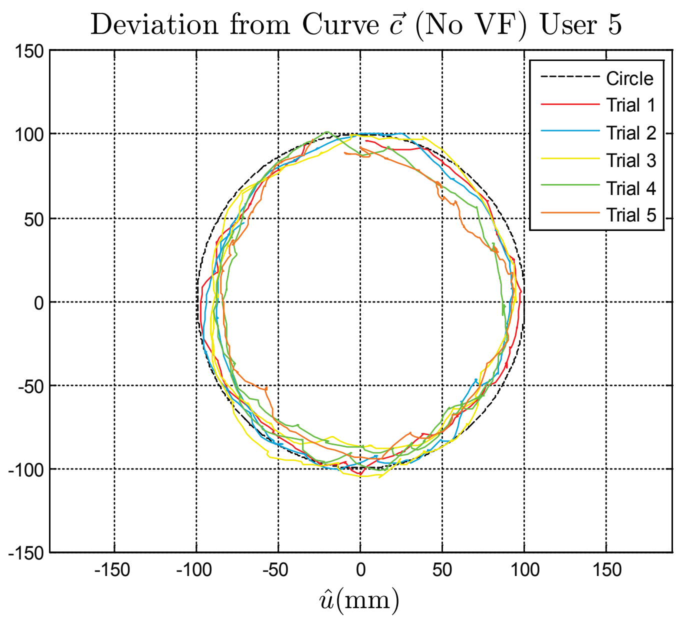
Follow Curve: With VF

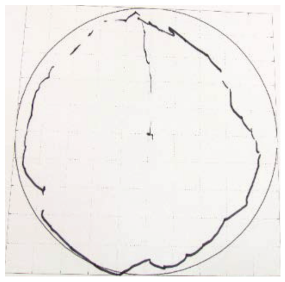




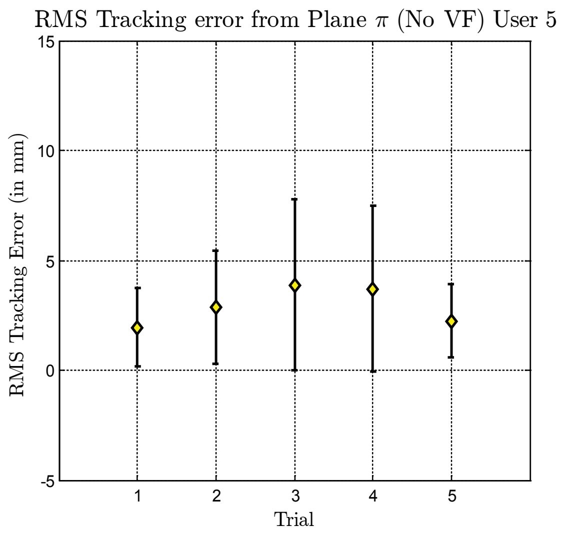
Virtual Fixture Task, Follow Curve

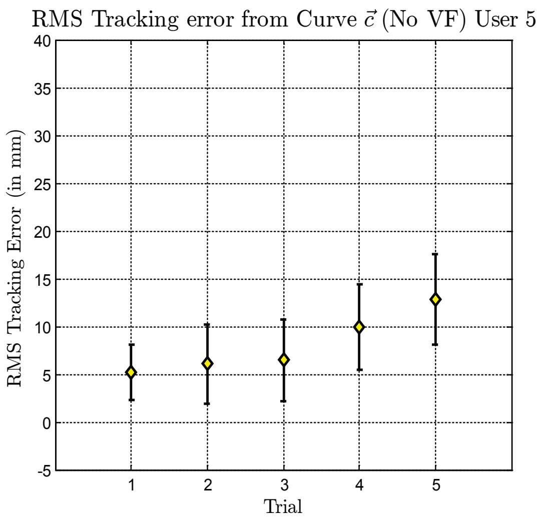
(e)User 5 (No VF)





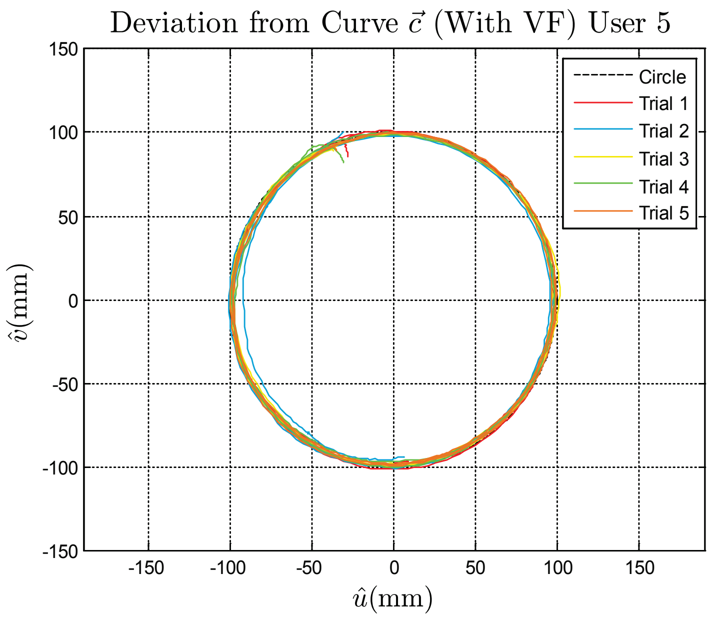
Follow Curve: No VF

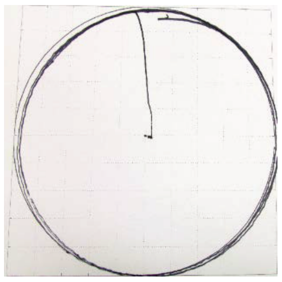


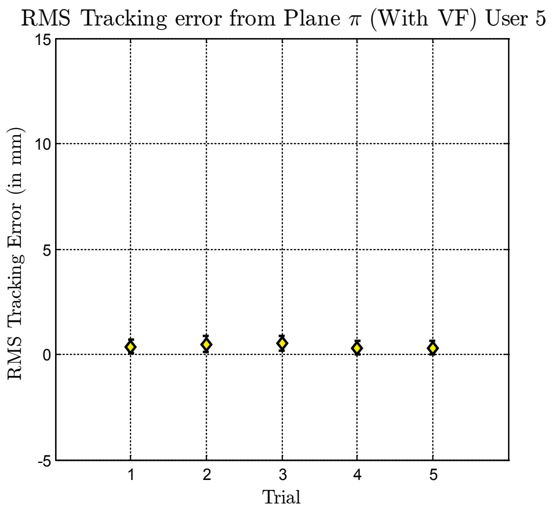


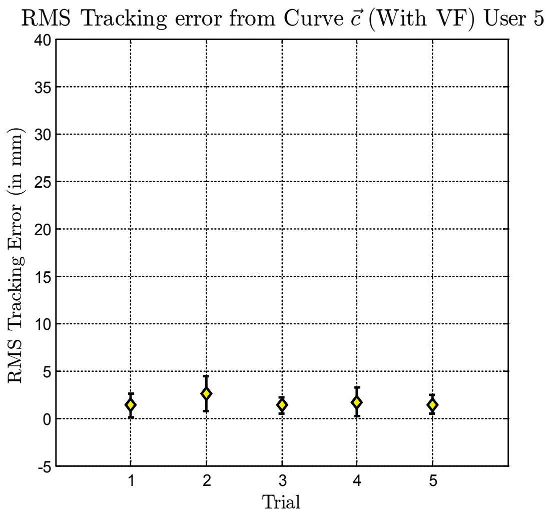
Virtual Fixture Task, Follow Curve

(e)User 5 (With VF)



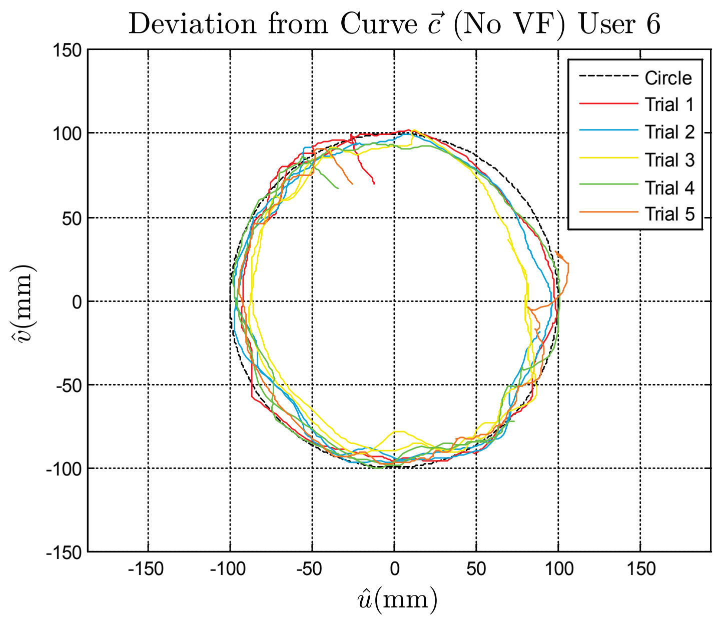

Follow Curve: With VF

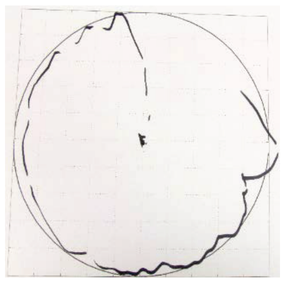




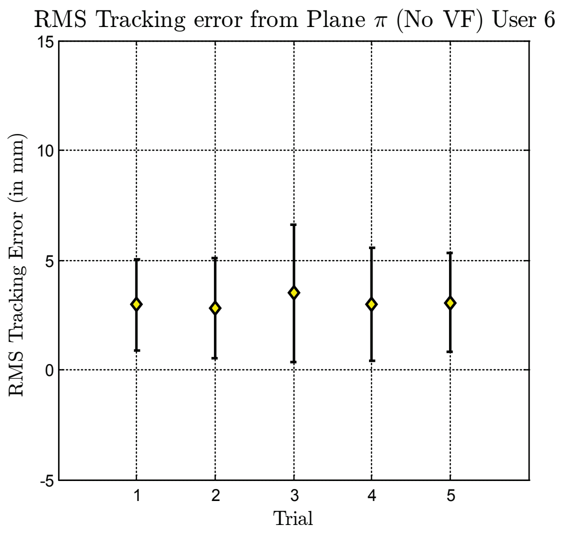
Virtual Fixture Task, Follow Curve

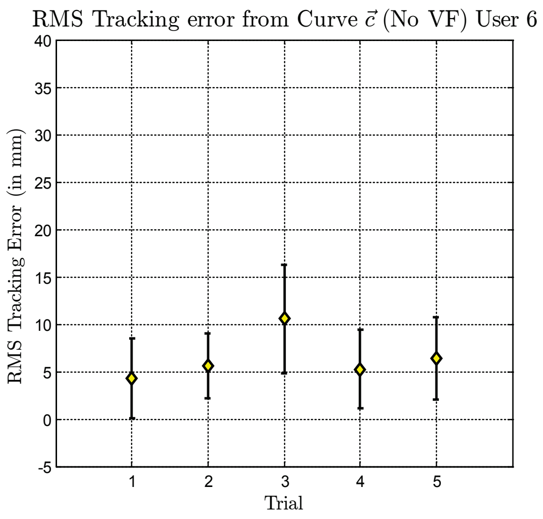
(f)User 6 (No VF)





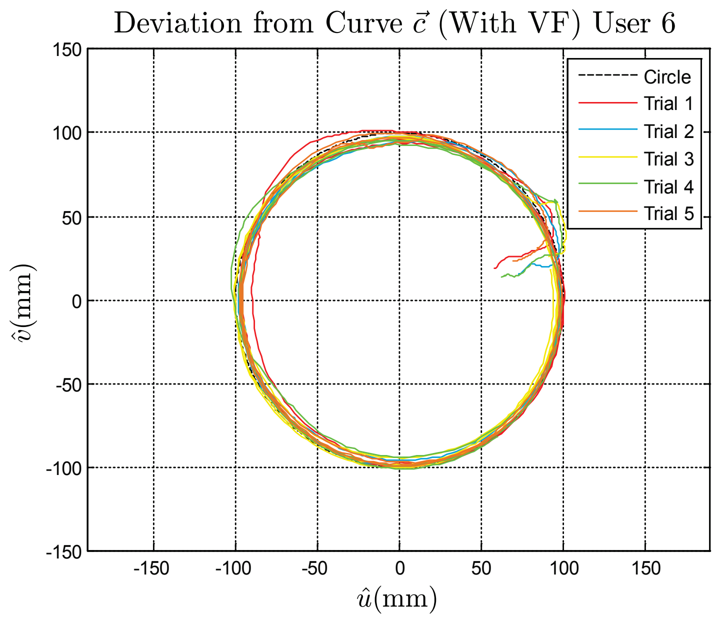
Follow Curve: No VF

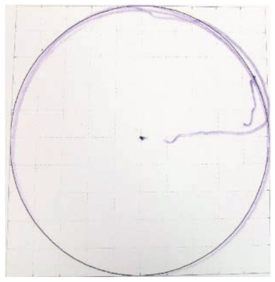




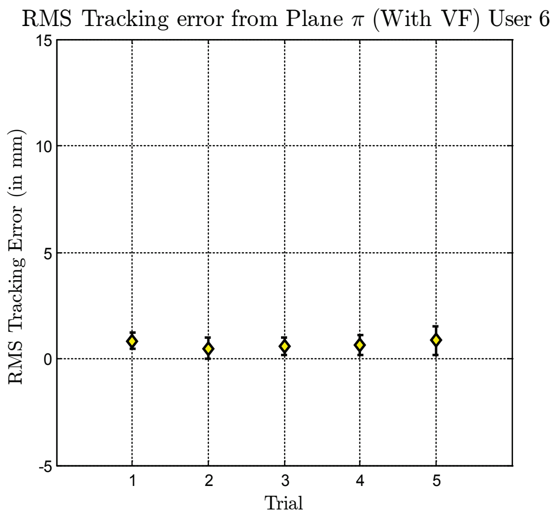
Virtual Fixture Task, Follow Curve

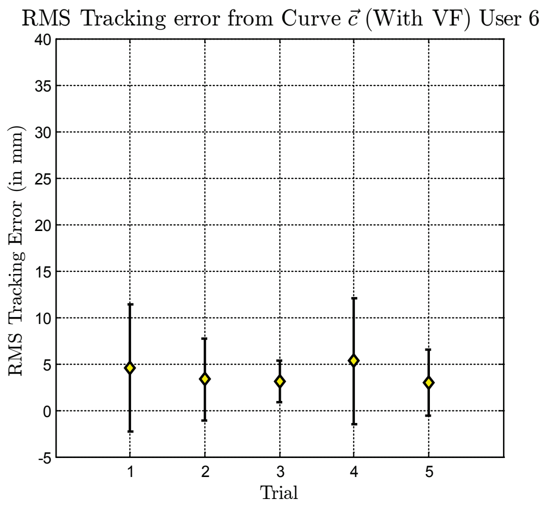
(f)User 6 (With VF)





Follow Curve: With VF





Virtual Fixture Task, Follow Curve

Paired t-test Results

	Without VF	With VF	Paired t-test	
User	$\bar{x}: mean\ deviations$	$\bar{y}: mean\ deviations$	p-value	Null-Hypothesis
User 1	3.479033	2.200798	0.014527< 0.05	Rejected
	7.143314	3.997630		
	5.170353	2.713022		
	4.853227	4.855025		
	6.715693	2.243710		
User 2	6.136941	5.241154	0.000292< 0.05	Rejected
	9.465244	2.566900		
	7.909974	2.024381		
	10.12944	2.653845		
	10.81166	2.020394		
User 3	37.98141	3.608272	0.002706< 0.05	Rejected
	14.80955	5.027399		
	33.96190	3.672182		
	47.75791	5.408982		
	33.69161	4.679667		

Table IV.3: Paired t-test Results for Follow Curve Task

Paired t-test Results

	Without VF	With VF	Paired t-test	
User	$\bar{x}$: *mean deviations*	$\bar{y}$: *mean deviations*	p-value	Null-Hypothesis
User 4	24.17575	2.768395	$0.011057 < 0.05$	Rejected
	8.879770	3.310666		
	10.11928	3.165263		
	14.75557	3.926037		
	10.55345	4.956641		
User 5	6.621142	1.631479	$0.002092 < 0.05$	Rejected
	8.232294	2.899722		
	9.355883	1.730830		
	12.68337	1.921520		
	14.55754	1.674700		
User 6	6.457845	5.194530	$0.008434 < 0.05$	Rejected
	7.704197	3.666168		
	13.16797	3.514628		
	7.469063	5.755594		
	8.676597	3.565781		

Paired t-test Results for Follow Curve Task

Virtual Fixture constraining movement within a curve

Methodology

The third task was to stay within a curve on a plane which is fixed with respect to the PUMA robot base.We measure the accuracy of the user in being able to stay within the curve and on the plane. Therefore, the two metrics considered are:

1. Mean deviation from the curve(only considering deviations outside the curve).
2. Mean deviation from the plane.

The combined metric considered for the paired t-test is,

$$\begin{aligned}\text{Weighed Deviation } &= \text{mean deviation from the curve} + \\ &\quad (0.75 * \text{mean deviation from the plane})\end{aligned}$$

For each user there are two vectors $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ containing five corresponding values of weighed deviations from the plane where,

$$\begin{aligned}\overline{\mathbf{x}} &: \text{weighed deviation from the plane without VF} \\ \overline{\mathbf{y}} &: \text{weighed deviation from the plane with VF}\end{aligned}$$

As in the previous two cases, the paired t-test is performed without assuming equal variances and considering the right tailed test. Therefore, the rejection of the null hypothesis would indicate a performance improvement with VF.

Figures IV.4 (a)-(e) present the results collected during the experiments for this task. The figures show the deviation of the robot end effector and the RMS tracking error from the plane $\boldsymbol{\pi}$.

Experimental Results

Figures IV.4 (a)-(e) show that without virtual fixture assistance, it is difficult for the user to perform the task. The user is able to keep the tool within the curve but not on the plane. However, with virtual fixture assistance it is fairly easy for the user to stay within the curve and on the plane. The paired t-test results (Table IV.4) also validate our observations, as the null hypothesis is rejected in every case. We can draw the same two conclusions as before:

1. The strong rejection of the null hypothesis at $p = 0.05$ tells us the that the two cases (with and without VF) are very different.
2. The null hypothesis was rejected in the right tailed paired t-test indicating that the alternative hypothesis is accepted. In our case the alternative hypothesis is that the mean deviations in the case with no VF is greater than the case with VF. This means that the case with VF is able to stay within the curve with much greater accuracy.

(a)User 1 (No VF)

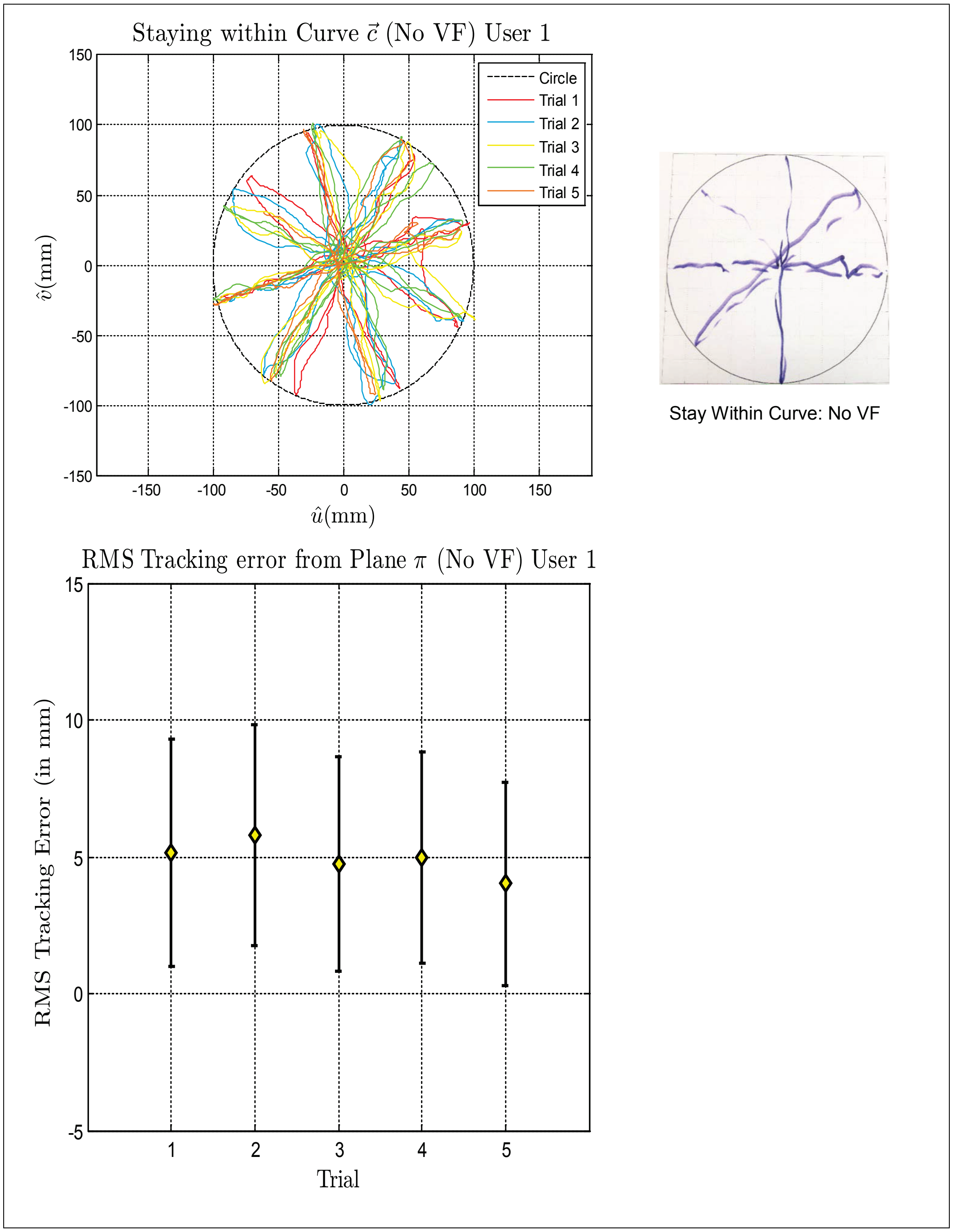
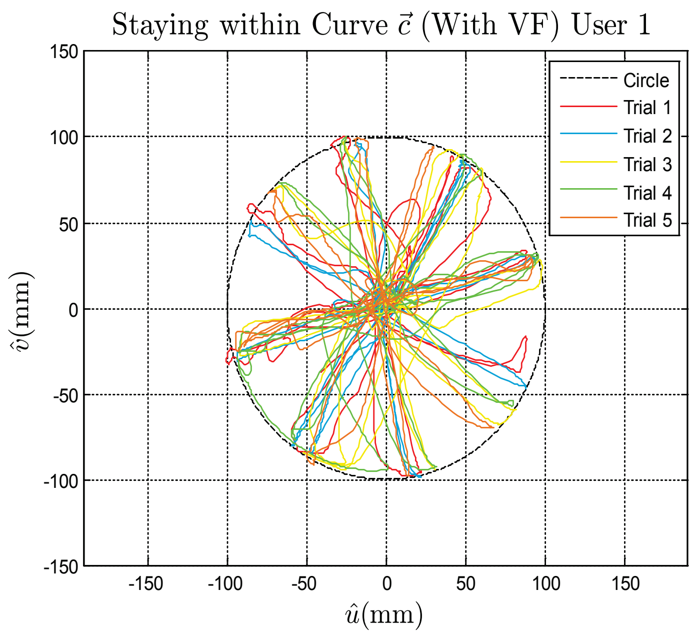
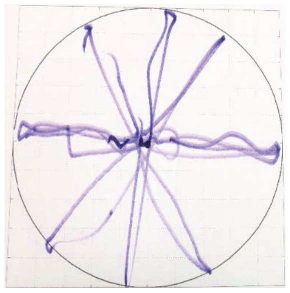
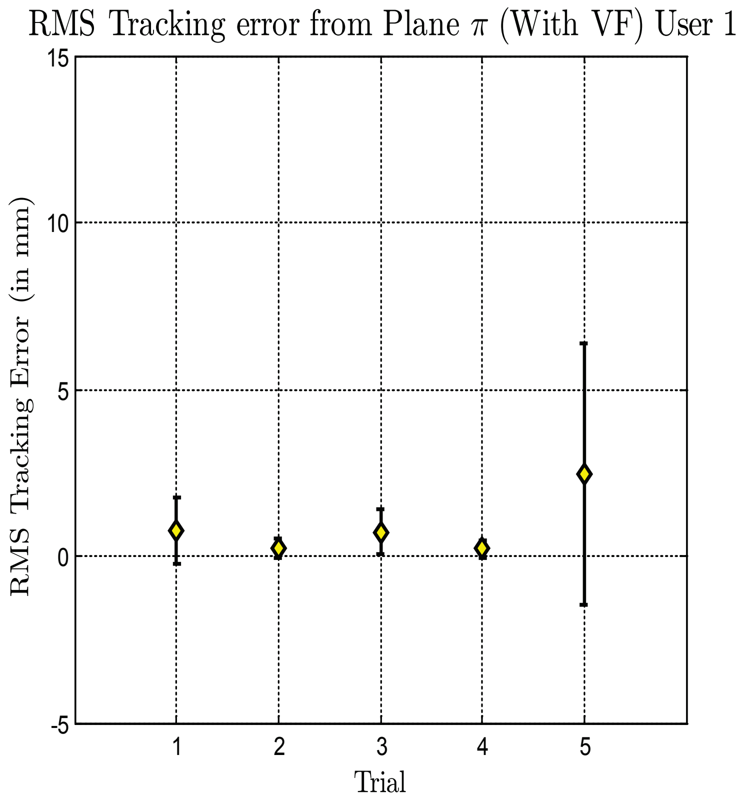


Figure IV.4: Virtual Fixture Task, Stay Within Curve

(a)User 1 (With VF)

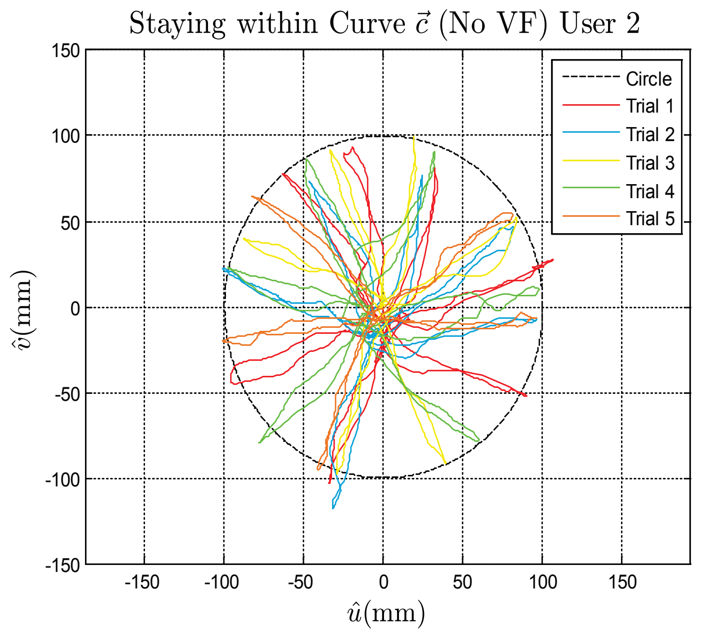


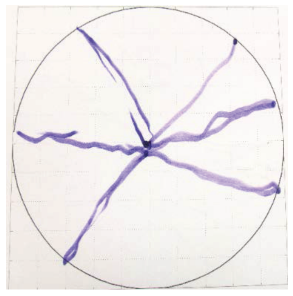

Stay Within Curve: With VF



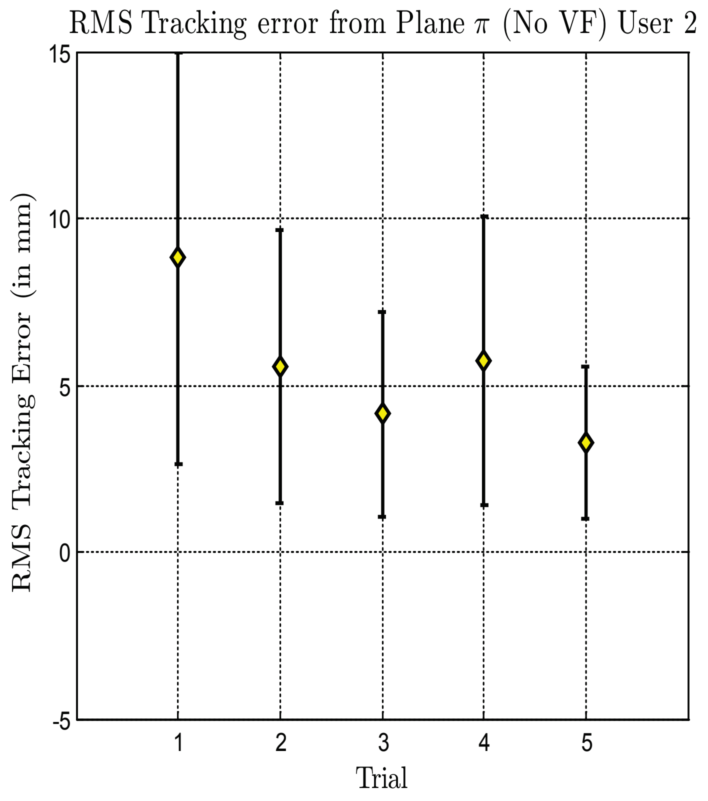
Virtual Fixture Task, Stay Within Curve

(b)User 2 (No VF)



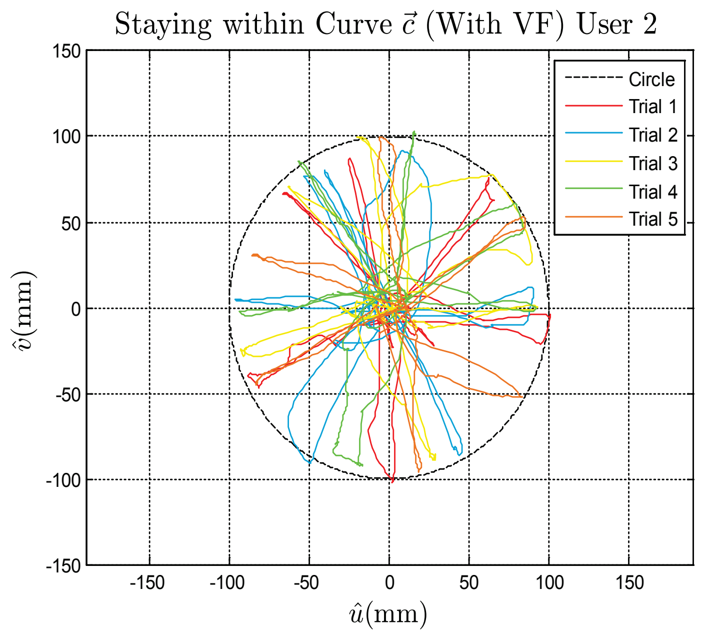


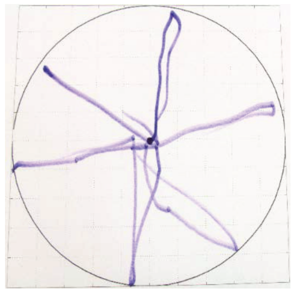
Stay Within Curve: No VF



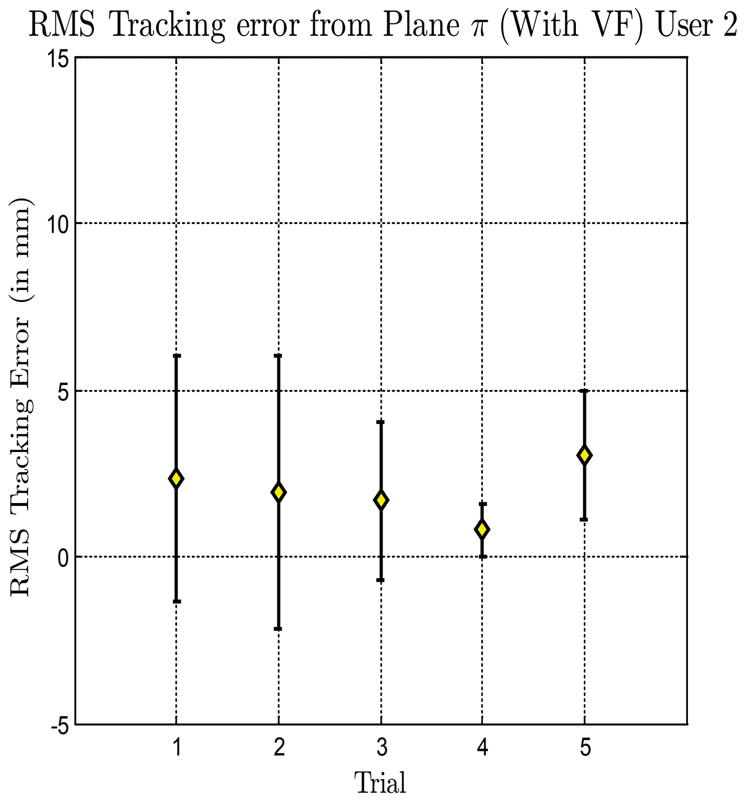
Virtual Fixture Task, Stay Within Curve

(b)User 2 (With VF)



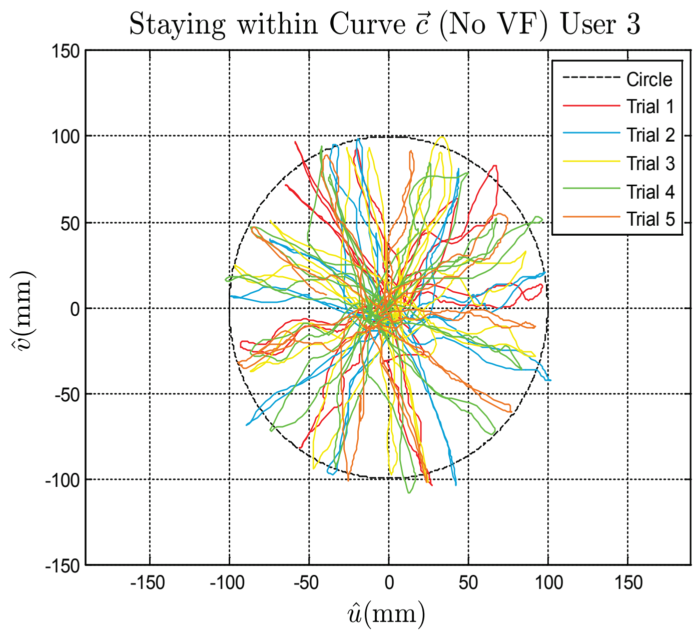


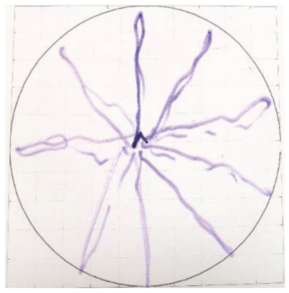
Stay Within Curve: With VF



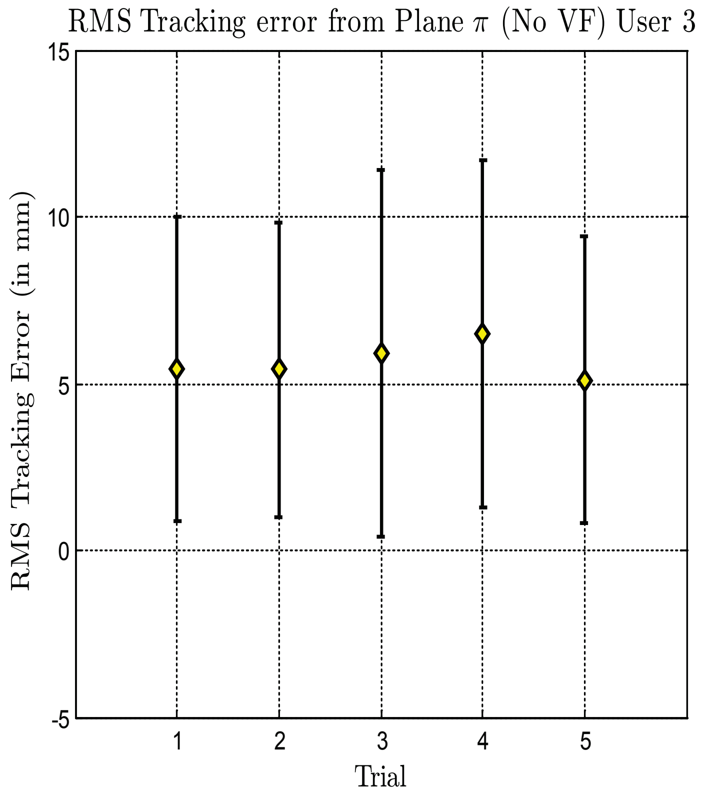
Virtual Fixture Task, Stay Within Curve

(c)User 3 (No VF)



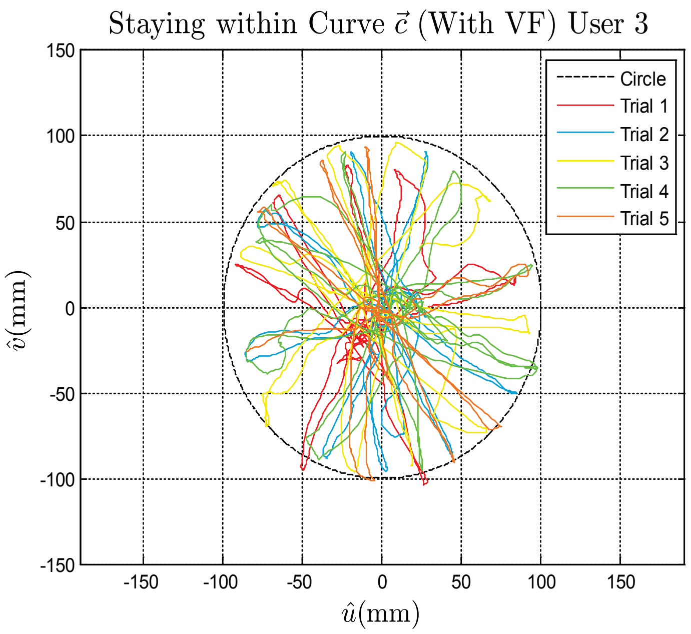


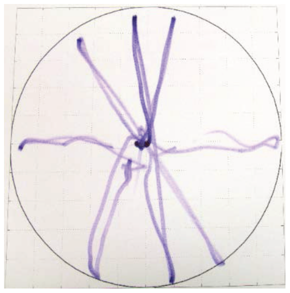
Stay Within Curve: No VF



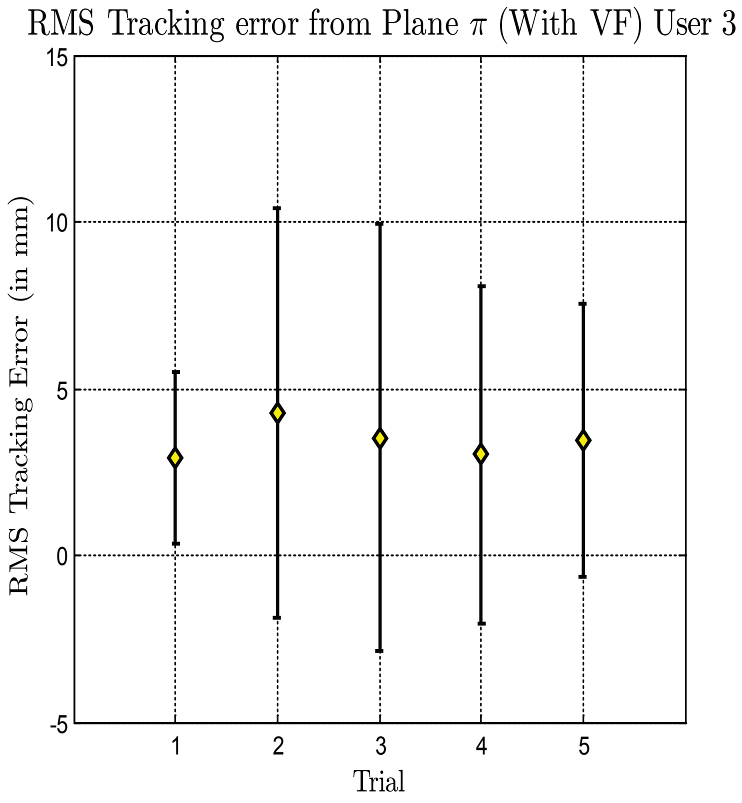
Virtual Fixture Task, Stay Within Curve

(c)User 3 (With VF)



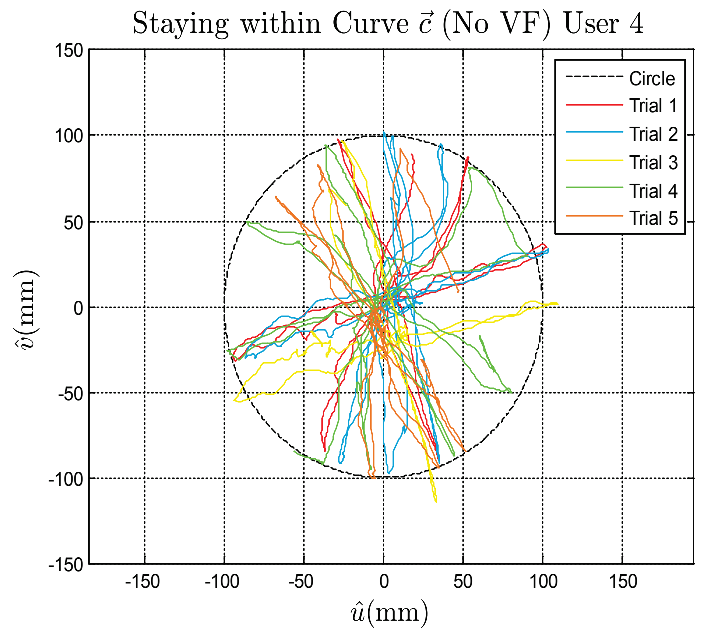


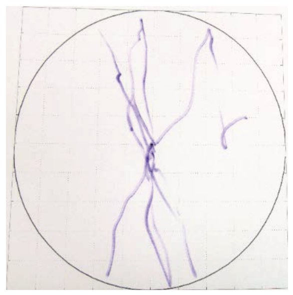
Stay Within Curve: With VF



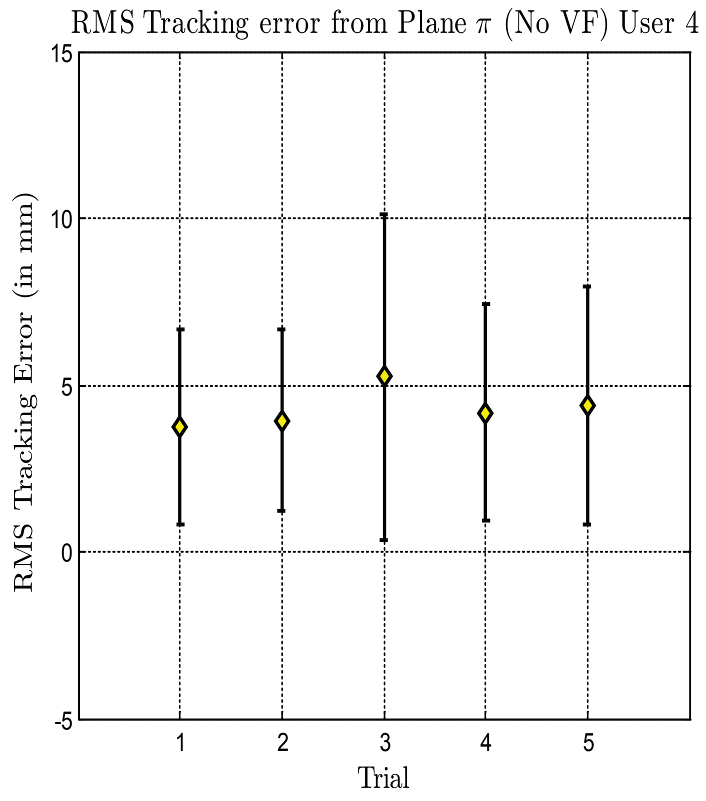
Virtual Fixture Task, Stay Within Curve

(d)User 4 (No VF)



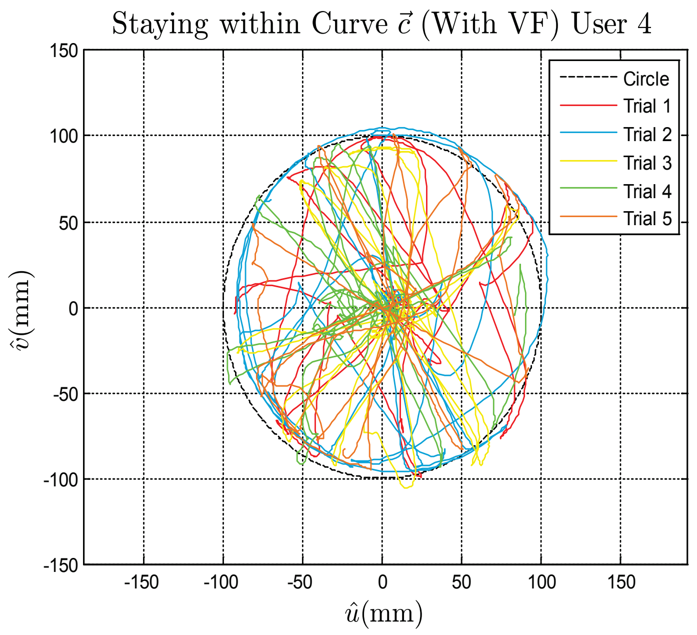


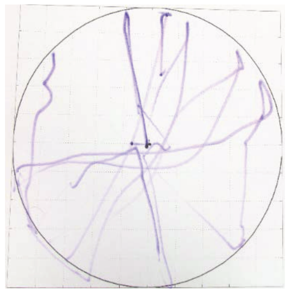
Stay Within Curve: No VF

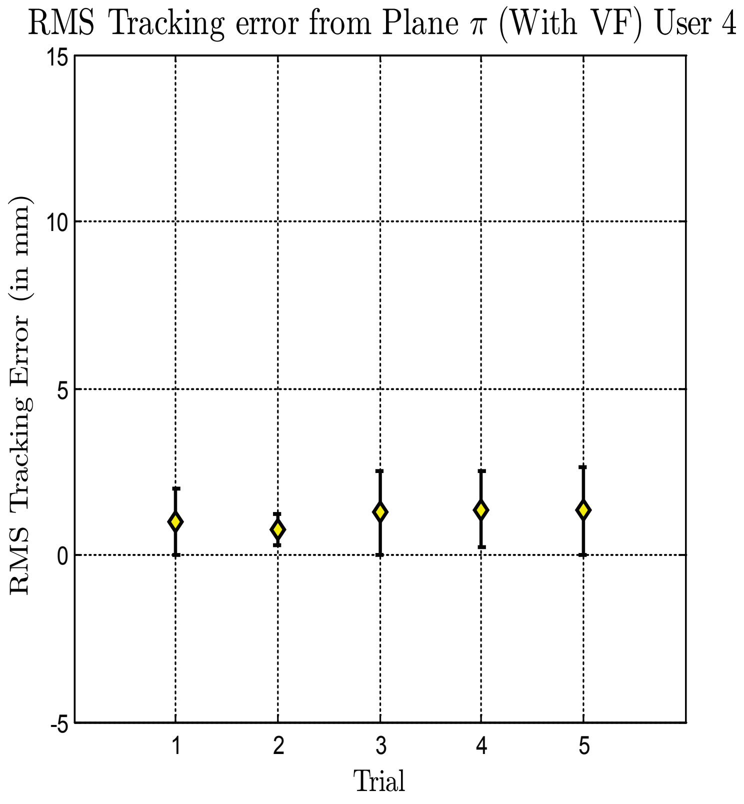


Virtual Fixture Task, Stay Within Curve

(d)User 4 (With VF)

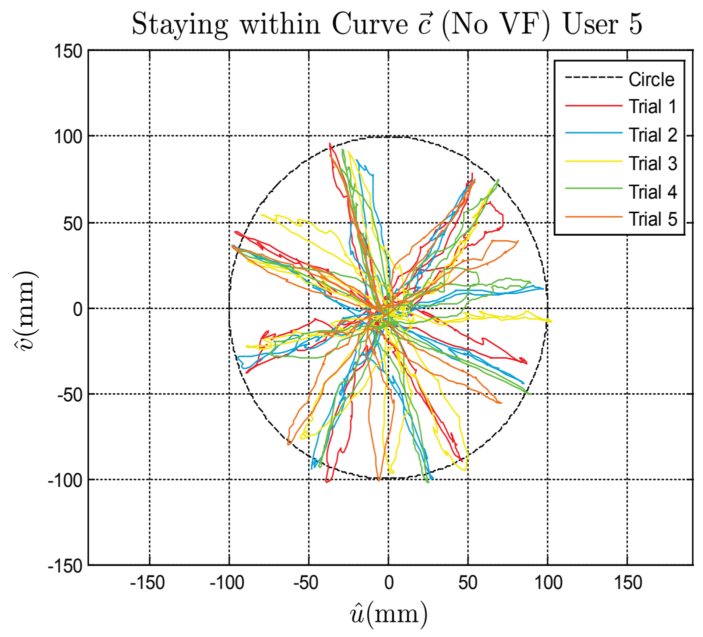


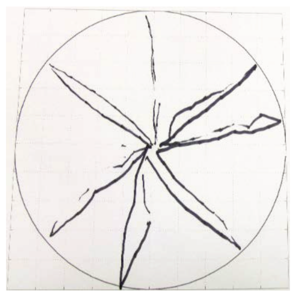

Stay Within Curve: With VF



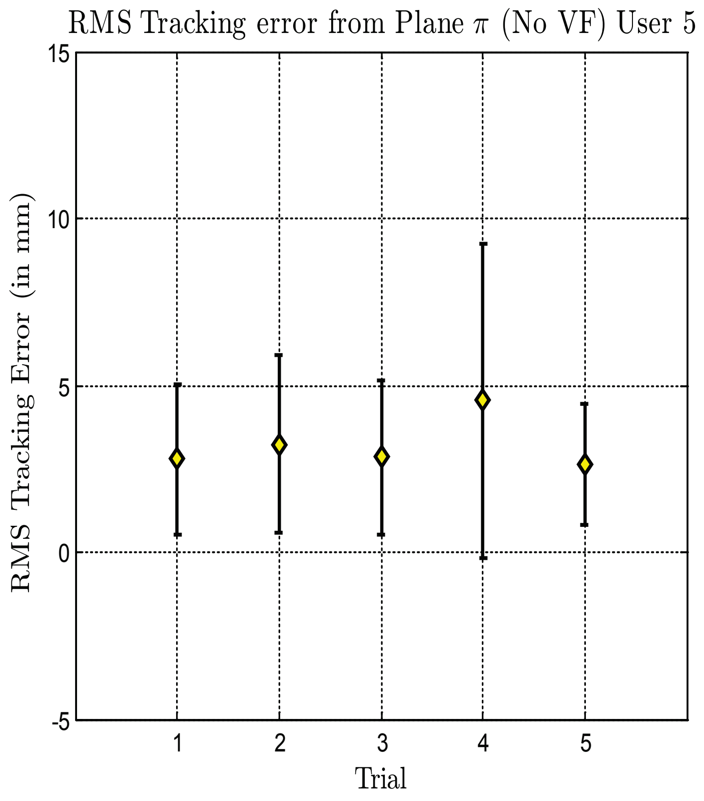
Virtual Fixture Task, Stay Within Curve

(e)User 5 (No VF)



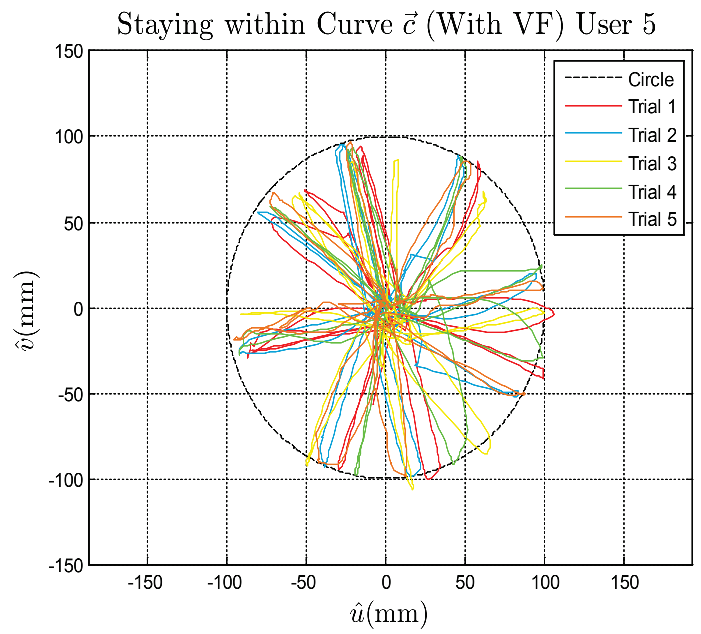


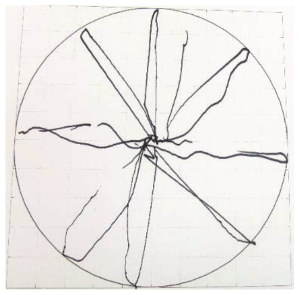
Stay Within Curve: No VF



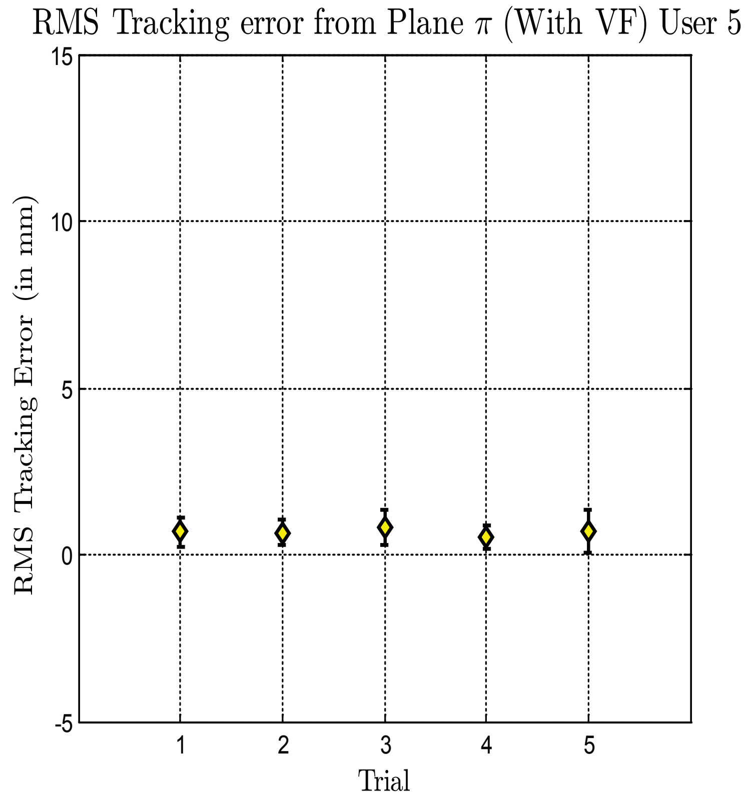
Virtual Fixture Task, Stay Within Curve

(e)User 5 (With VF)



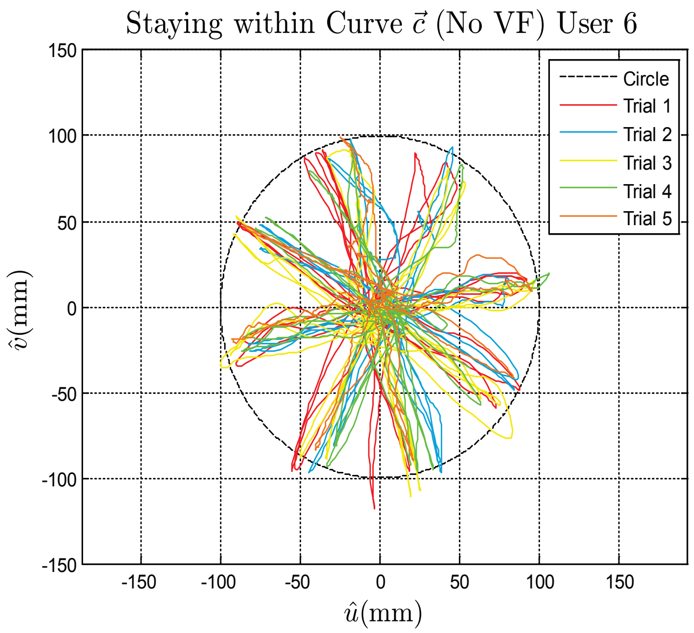


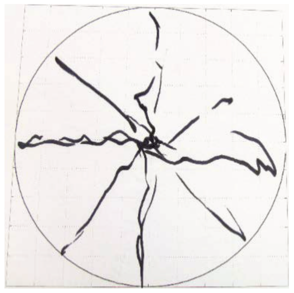
Stay Within Curve: With VF



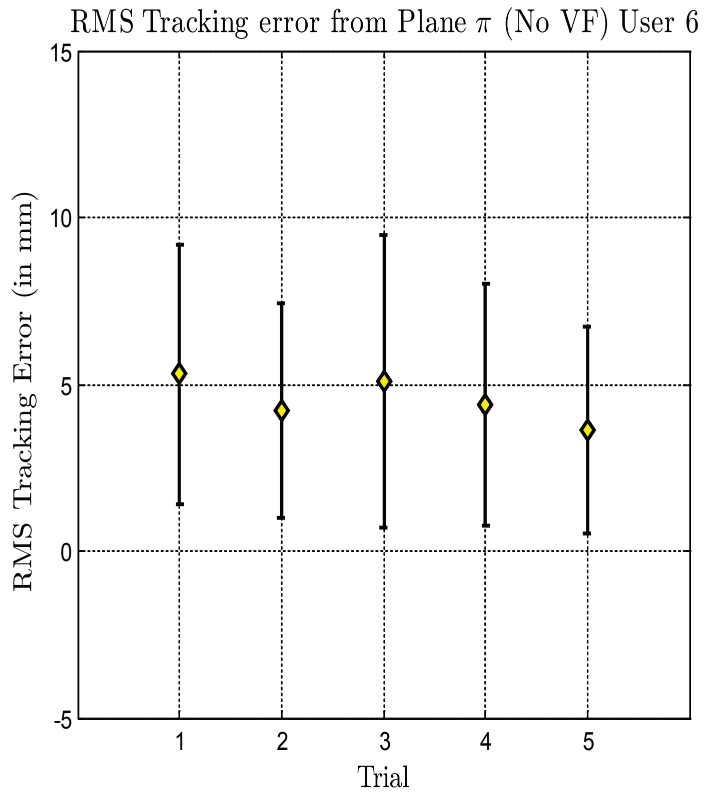
Virtual Fixture Task, Stay Within Curve

(f)User 6 (No VF)



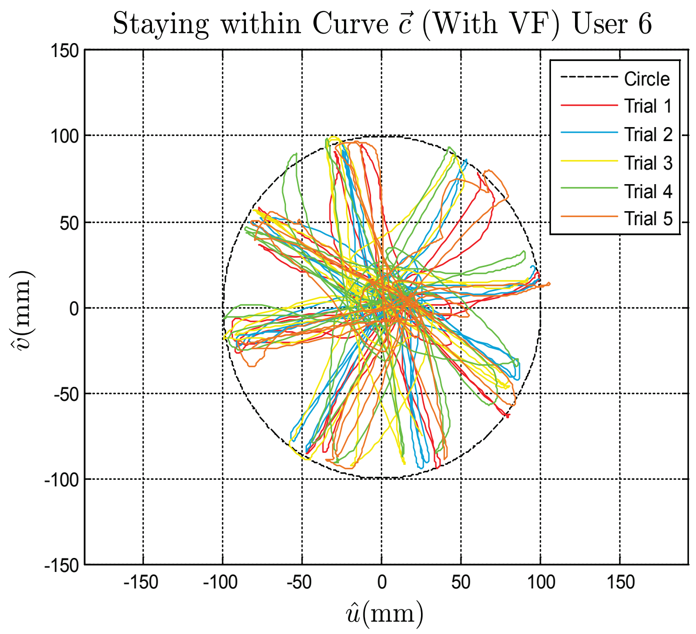


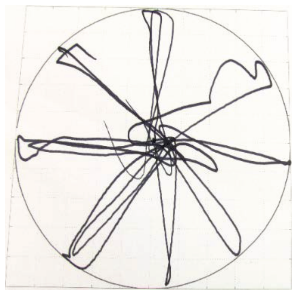
Stay Within Curve: No VF



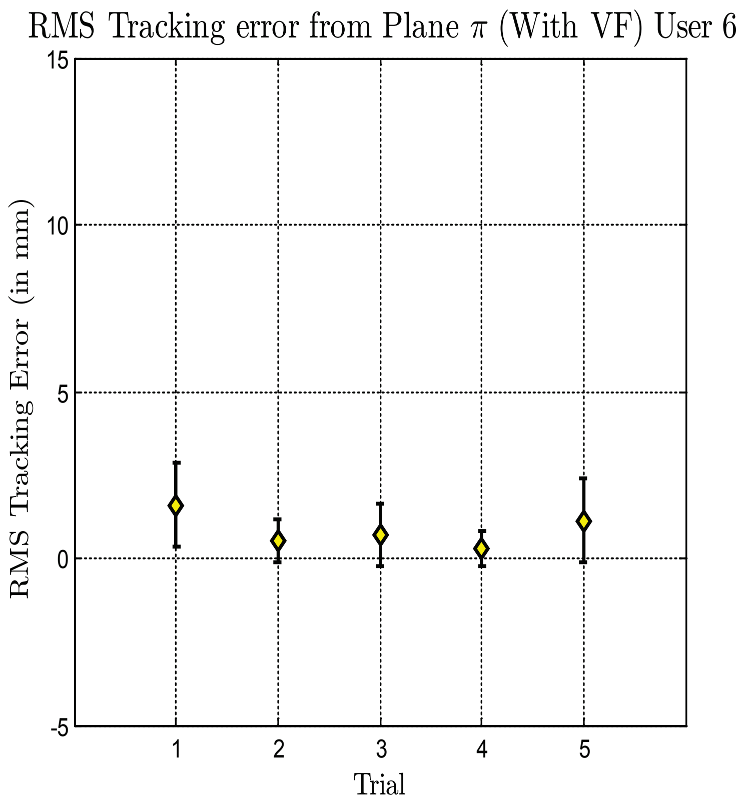
Virtual Fixture Task, Stay Within Curve

(f)User 6 (With VF)





Stay Within Curve: With VF



Virtual Fixture Task, Stay Within Curve

Paired t-test Results

	Without VF	With VF	Paired t-test	
User	$\bar{x}: mean\ deviations$	$\bar{y}: mean\ deviations$	p-value	Null-Hypothesis
User 1	3.862043	0.759550	0.000039< 0.05	Rejected
	4.373461	0.176768		
	3.648918	0.632341		
	3.780079	0.258398		
	3.052170	1.883143		
User 2	6.922211	1.796048	0.008999< 0.05	Rejected
	4.536255	1.476640		
	3.111896	1.299741		
	4.416088	0.816085		
	2.578517	2.279871		
User 3	4.241693	2.527077	0.000056 < 0.05	Rejected
	4.355774	3.217793		
	4.477029	2.658185		
	4.986963	2.377743		
	3.859368	2.689639		

Table IV.4: Paired t-test Results for Stay Within Curve Task

Paired t-test Results

	Without VF	With VF	Paired t-test	
User	$\bar{x}: mean\ deviations$	$\bar{y}: mean\ deviations$	p-value	Null-Hypothesis
User 4	2.914674	1.230420	$0.000524 < 0.05$	Rejected
	3.093142	1.084375		
	4.412812	1.370613		
	3.143789	1.175740		
	3.289693	1.190544		
User 5	2.200119	0.767832	$0.000681 < 0.05$	Rejected
	2.494787	0.493995		
	2.178701	0.776349		
	3.483535	0.449716		
	2.051492	0.541929		
User 6	4.126676	1.237784	$0.000021 < 0.05$	Rejected
	3.230082	0.408064		
	3.991948	0.622815		
	3.329132	0.287757		
	2.728243	1.045577		

Paired t-test Results for Stay Within Curve Task

IV.2 Conclusion

The experiments conducted validated the derivation of the virtual fixture laws. An important point to note is that the performance with virtual fixture assistance is dependent on the tuning of the gains for the virtual fixture laws. An accurate tuning of the parameters affects the results of the experiments.

In the task where the user had to follow a plane, both the plots and the paired t-test results clearly indicate the performance improvement with virtual fixture assistance. Another important point to consider is the reduction of cognitive load for the user. Therefore, the user is able to perform the task with greater accuracy and reduced cognitive load with virtual fixture assistance. In the second task, the user had to follow a curve on a plane. Again, the plots and the paired t-test results indicate a visible improvement with VF assistance. This task required the user to satisfy the constraints of keeping the tool on the plane as well as the curve. This resulted in an even greater cognitive load than the previous case. However, we observed that with VF assistance the user was not only able to complete the task with ease but also complete the task multiple times in the assigned time. The third task required the user to stay within a curve on the plane. The plots indicate that the user is able to stay within the curve with ease even without virtual fixture assistance. However, the deviations from the plane indicate that the user was able to keep the tool on the plane with greater ease in the case with virtual fixture assistance.

The strong rejection of the null hypothesis in all three tasks implies that virtual fixture assistance definitely improves the performance of the user.

CHAPTER V

CONCLUSIONS

In this thesis, assistive manipulation algorithms suitable for both cooperative manipulation and telemanipulation were presented. We derived our virtual fixture laws based on the theory of projections and extended the approach to derive a VF law which constrains the motion of the robot end effector within a curve. A library of virtual fixture primitives was implemented which would allow definitions of new virtual fixtures. Experiments were conducted to validate the derived virtual fixture laws and the results show that VF assistance significantly improves the performance of the users. We also proposed the concept of user-specified virtual fixtures which would allow the specification of motion constraints intraoperatively. Most of the previous approaches are not applicable to highly deformable anatomy such as the bladder. We can simplify this problem by letting the surgeon specify the region of operation intraoperatively by tracing a closed path with a visible spectrum laser.

The utility of the virtual fixture approaches presented in this thesis would truly be realized once we integrate our algorithms with a setup that would let us specify the constraint curve in real-time. Future work includes design and implementation of a setup that would allow visual registration of surfaces using laser structured light (Figure V.1). The laser structured light would be used to generate a grid of points on the surface where we want to define the constraint curve. Once the surface is registered, the user would be able to guide a visible spectrum laser in the camera view. We would segment out the path of the laser using optical flow algorithms

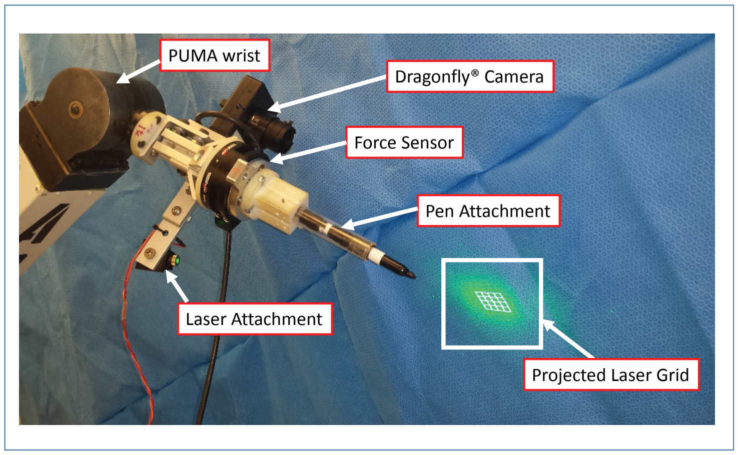


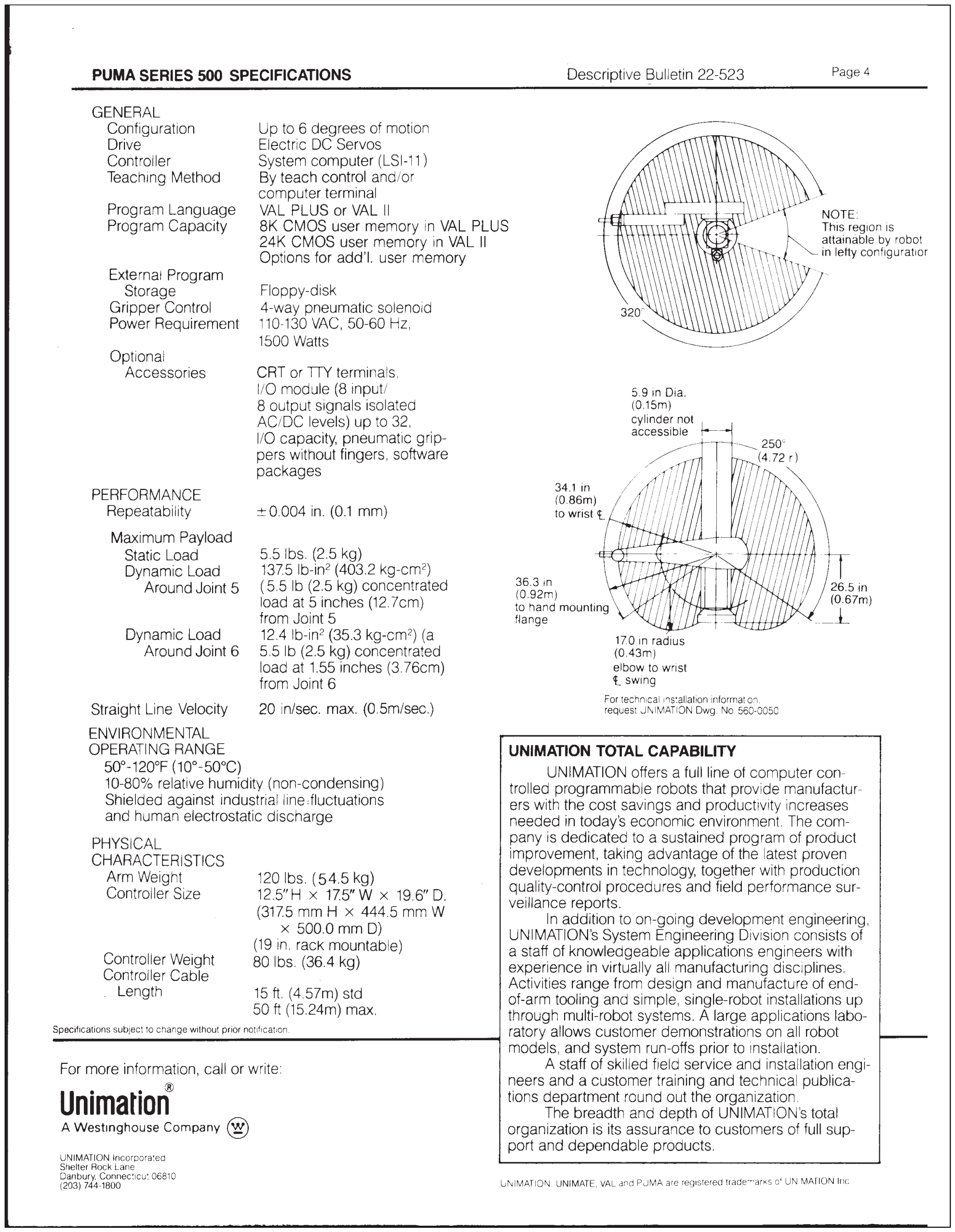
Figure V.1: Experimental Setup for validating user-specified VF based on vision

and generate the point-cloud representing the curve on the surface. Once we have the representation of the curve, we can easily implement the virtual fixture laws implemented in this thesis.

Appendices

APPENDIX A

PUMA SPECIFICATION SHEET



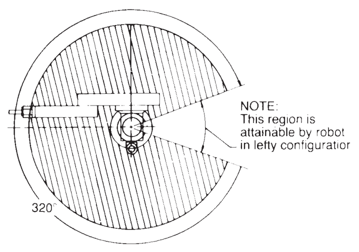
PUMA SERIES 500 SPECIFICATIONS Descriptive Bulletin 22-523 Page 4

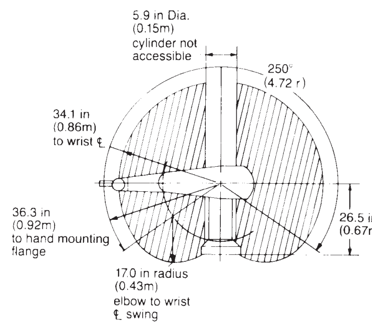
GENERAL	
Configuration	Up to 6 degrees of motion
Drive	Electric DC Servos
Controller	System computer (LSI-11)
Teaching Method	By teach control and/or computer terminal
Program Language	VAL PLUS or VAL II
Program Capacity	8K CMOS user memory in VAL PLUS 24K CMOS user memory in VAL II Options for add'l. user memory
External Program Storage	Floppy-disk
Gripper Control	4-way pneumatic solenoid
Power Requirement	110-130 VAC, 50-60 Hz, 1500 Watts
Optional Accessories	CRT or TTY terminals, I/O module (8 input/8 output signals isolated AC/DC levels) up to 32, I/O capacity, pneumatic grippers without fingers, software packages
PERFORMANCE	
Repeatability	± 0.004 in. (0.1 mm)
Maximum Payload	
Static Load	5.5 lbs. (2.5 kg)
Dynamic Load Around Joint 5	137.5 lb-in^2 (403.2 kg-cm^2) (5.5 lb (2.5 kg) concentrated load at 5 inches (12.7cm) from Joint 5
Dynamic Load Around Joint 6	12.4 lb-in^2 (35.3 kg-cm^2) (a 5.5 lb (2.5 kg) concentrated load at 1.55 inches (3.76cm) from Joint 6
Straight Line Velocity	20 in/sec. max. (0.5m/sec.)

ENVIRONMENTAL OPERATING RANGE
50°-120°F (10°-50°C)
10-80% relative humidity (non-condensing)
Shielded against industrial line fluctuations and human electrostatic discharge

PHYSICAL CHARACTERISTICS	
Arm Weight	120 lbs. (54.5 kg)
Controller Size	12.5″H × 17.5″W × 19.6″D. (317.5 mm H × 444.5 mm W × 500.0 mm D) (19 in. rack mountable)
Controller Weight	80 lbs. (36.4 kg)
Controller Cable Length	15 ft. (4.57m) std 50 ft (15.24m) max.

Specifications subject to change without prior notification.


NOTE: This region is attainable by robot in lefty configuration
320°


5.9 in Dia. (0.15m) cylinder not accessible
250° (4.72 r)
34.1 in (0.86m) to wrist ℄
36.3 in (0.92m) to hand mounting flange
26.5 in (0.67m)
17.0 in radius (0.43m) elbow to wrist ℄ swing

For technical installation information, request UNIMATION Dwg. No. 560-0050

UNIMATION TOTAL CAPABILITY

UNIMATION offers a full line of computer controlled programmable robots that provide manufacturers with the cost savings and productivity increases needed in today's economic environment. The company is dedicated to a sustained program of product improvement, taking advantage of the latest proven developments in technology, together with production quality-control procedures and field performance surveillance reports.

In addition to on-going development engineering, UNIMATION's System Engineering Division consists of a staff of knowledgeable applications engineers with experience in virtually all manufacturing disciplines. Activities range from design and manufacture of end-of-arm tooling and simple, single-robot installations up through multi-robot systems. A large applications laboratory allows customer demonstrations on all robot models, and system run-offs prior to installation.

A staff of skilled field service and installation engineers and a customer training and technical publications department round out the organization.

The breadth and depth of UNIMATION's total organization is its assurance to customers of full support and dependable products.

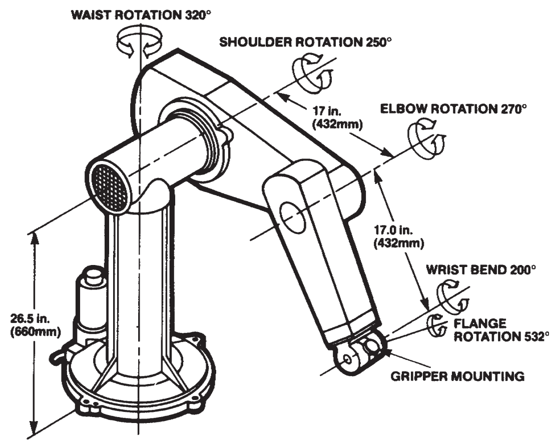
For more information, call or write:

Unimation®
A Westinghouse Company

UNIMATION Incorporated
Shelter Rock Lane
Danbury, Connecticut 06810
(203) 744-1800

UNIMATION, UNIMATE, VAL and PUMA are registered trademarks of UNIMATION Inc.

UNIMATE PUMA
500 Series



Performance

REPEATABILITY	±0.004 in. (±0.1 mm)
LOAD CAPACITY	5.5 lbs. (2.5 Kg)
STRAIGHT LINE VELOCITY	20 in/s max. (0.5 m/s max.)
ENVIRONMENTAL	50-120° F (10-50° C)
REQUIREMENTS	80% humidity (non-condensing). Shielded against industrial line fluctuations and human electro-static discharge

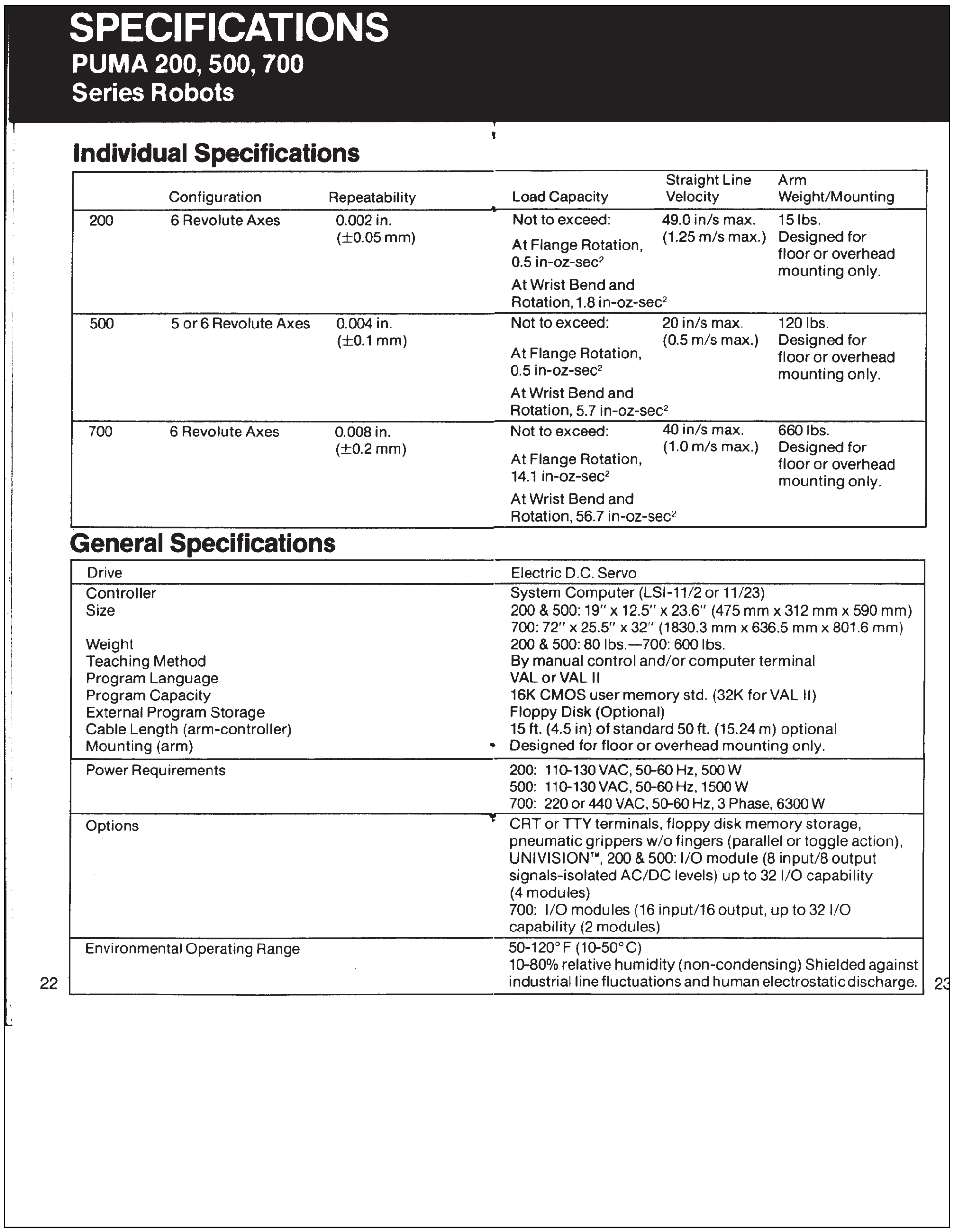
Physical Characteristics

ARM WEIGHT	120 lbs. (54.5 Kg)
CONTROLLER SIZE	19″ x 12.5″ x 23.6″ (475 mm x 312.5 mm x 590 mm)
CONTROLLER WEIGHT	80 lbs. (176 Kg)
CONTROLLER CABLE LENGTH	15 ft. (4.5 m) Standard 50 ft. (15 m) Optional

General Specification

CONFIGURATION	5 revolute axes or 6 revolute axes
DRIVE	Electric DC Servo
CONTROLLER	System Computer (LSI-11/2 or 11/23)
Teaching Method	By manual control and/or computer terminal
Program Language	VAL or VAL II
Program Capacity	16K CMOS user memory std. (32K for VAL II)
External Program Storage	Floppy-disk (optional)
GRIPPER CONTROL	4-way pneumatic solenoid
POWER REQUIREMENT	110-130 V AC 50-60 Hz, 1500 W
OPTIONAL ACCESSORIES	CRT or TTY terminals, floppy-disk memory storage, I/O module, 8 input/8 output signals (max. 32)—isolated AC/DC levels. Pneumatic gripper w/o fingers

1



SPECIFICATIONS

PUMA 200, 500, 700
Series Robots

Individual Specifications

	Configuration	Repeatability	Load Capacity	Straight Line Velocity	Arm Weight/Mounting
200	6 Revolute Axes	0.002 in. (±0.05 mm)	Not to exceed: At Flange Rotation, 0.5 in-oz-sec^2 At Wrist Bend and Rotation, 1.8 in-oz-sec^2	49.0 in/s max. (1.25 m/s max.)	15 lbs. Designed for floor or overhead mounting only.
500	5 or 6 Revolute Axes	0.004 in. (±0.1 mm)	Not to exceed: At Flange Rotation, 0.5 in-oz-sec^2 At Wrist Bend and Rotation, 5.7 in-oz-sec^2	20 in/s max. (0.5 m/s max.)	120 lbs. Designed for floor or overhead mounting only.
700	6 Revolute Axes	0.008 in. (±0.2 mm)	Not to exceed: At Flange Rotation, 14.1 in-oz-sec^2 At Wrist Bend and Rotation, 56.7 in-oz-sec^2	40 in/s max. (1.0 m/s max.)	660 lbs. Designed for floor or overhead mounting only.

General Specifications

Drive	Electric D.C. Servo
Controller	System Computer (LSI-11/2 or 11/23)
Size	200 & 500: 19″ x 12.5″ x 23.6″ (475 mm x 312 mm x 590 mm) 700: 72″ x 25.5″ x 32″ (1830.3 mm x 636.5 mm x 801.6 mm)
Weight	200 & 500: 80 lbs.—700: 600 lbs.
Teaching Method	By manual control and/or computer terminal
Program Language	VAL or VAL II
Program Capacity	16K CMOS user memory std. (32K for VAL II)
External Program Storage	Floppy Disk (Optional)
Cable Length (arm-controller)	15 ft. (4.5 in) of standard 50 ft. (15.24 m) optional
Mounting (arm)	Designed for floor or overhead mounting only.
Power Requirements	200: 110-130 VAC, 50-60 Hz, 500 W 500: 110-130 VAC, 50-60 Hz, 1500 W 700: 220 or 440 VAC, 50-60 Hz, 3 Phase, 6300 W
Options	CRT or TTY terminals, floppy disk memory storage, pneumatic grippers w/o fingers (parallel or toggle action), UNIVISION™, 200 & 500: I/O module (8 input/8 output signals-isolated AC/DC levels) up to 32 I/O capability (4 modules) 700: I/O modules (16 input/16 output, up to 32 I/O capability (2 modules)
Environmental Operating Range	50-120°F (10-50°C) 10-80% relative humidity (non-condensing) Shielded against industrial line fluctuations and human electrostatic discharge.

22

23

APPENDIX B

VIRTUAL FIXTURE CODE

Listing B.1: Function to compute the closest point to the curve

```

%% Function Description
% Date Created: 02/25/2014
% Created By: Aditya Bhowmick
%---------------------------------------------------------------
% Description: Function to compute distance to curve and plane
%---------------------------------------------------------------
% Last Edited: Aditya Bhowmick
% Edited on: 03/17/2014

function [u, v, local_curve_tangent, inside_curve, min_distance, d1, d2]...
                              = Compute_closest_point(a,p,target_curve,...
                                                          X_pi, r_curve)
% Defining variables
%% Getting index of minimum point
% Getting the number of points in the curve
num_points=length(target_curve);

% Making sure that points are given in columns
if size(target_curve,1)>size(target_curve,2) % points are given in rows
    target_curve=target_curve';
end;
```

```

% Getting vectors from p to points on the target_curve
matrix_p_to_curve_vecs=(target_curve-p(:)*ones(1,num_points));

% Getting distance^2 of each of the points on the curve from p
distance_row_vec=sum(matrix_p_to_curve_vecs.*matrix_p_to_curve_vecs,1);

% Minimal distance to the curve
[min_distance,index] = min(distance_row_vec);


%% Computing the local tangent
% local_curve_tangent = zeros(size(target_curve,1),1);
if (index>1) && (index<num_points)
    local_curve_tangent=(target_curve(:,index+1)-...
                         target_curve(:,index-1))/...
                         norm(target_curve(:,index+1)...
                              -target_curve(:,index-1));
elseif index==1
    local_curve_tangent=(target_curve(:,index+1)-...
                         target_curve(:,num_points))/...
                         norm(target_curve(:,index+1)...
                              -target_curve(:,num_points));
else
    local_curve_tangent=(target_curve(:,1)-...
                         target_curve(:,num_points-1))...
                             /norm(target_curve(:,1)...
                            -target_curve(:,num_points-1));
end;

```

```
% Saving the closest point on the curve to point p
closest_point=target_curve(:,index);

%% Computing vector pointing to the closest point on the curve
v=(closest_point-p(:));

%% Computing minimum distance to plane
% Compute projection matrix
P_pi = X_pi*pinv(X_pi'*X_pi)*X_pi';

% Getting null space projection of this matrix
null_P_pi = eye(3,3) - P_pi;

% Computing the closest distance to the plane
u = null_P_pi * (a(:) - p(:));

%% Computing if end effector is inside the curve

% Computing distance d1 between closest point on curve to current start
% position and the origin(= p_slave_start = a)
d1 = r_curve;

% Computing distance d2 between current position of end effector and the
% origin (= p_slave_start = a)
d2 = norm(p-a);

if d1 >= d2
    inside_curve = 1;
else
    inside_curve = 0;
```

```
end
%% Computing minimum distance to the curve
min_distance = d1 - d2;
```

Listing B.2: Function to apply the VF laws

```
%% Function Description
% Date Created: 03/11/2014
% Created By: Aditya Bhowmick

%-----------------------------------------------------------------
% Description: Function applying virtual fixture on the end effector of the
% PUMA560. There are three VF_modes which are:-
% 1) Mode 1:- Virtual fixture on a plane
% 2) Mode 2:- Virtual fixture on a curve in a plane.
% 3) Mode 3:- Virtual fixture to stay inside a curve on a plane
% Edited from original function given by Nabil

% Force Feedback
% The force feedback is computed on the basis of u or v depending on the
% mode that we are in.
%-----------------------------------------------------------------

% Last Edited: Aditya Bhowmick
% Edited on: 04/17/2014

function [computed_twist, computed_force,...
                 k_comp,k_tau_computed, distance_from_plane] = ...
                        apply_VF(VF_mode, enable_FF, ...
                               twist, enable_motion,u, v, curve_tangent,...
                               min_distance,...
                               X_pi, r_curve, scale_r, tanh_gain,...
                               ka, k_tau, kp_pi, kp_c,...
                               epsilon_pi, epsilon_c,...
                               K_stiffness, B_damping, R_rob2omni)
```

```

%---------------------------------------------------------------
% We want to find the projection of the pose twist along the curve_tangent
% and the plane
t = curve_tangent;

% Extracting just the pose twist because we do not want to change the
% orientation twist
pose_twist = twist(1:3);

% Setting distance threshold for applying the VF law
epsilon_plane = 0.01;

% Setting parameter to allow faster movement away from the plane
k_tau_out_in_ratio=2;

% Defining hard coded plane normal
plane_normal = [0 0 1];

% Computing distance from plane for applying VF. We need to make sure that
% this is a signed distance.
distance_from_plane = -dot(u,plane_normal);

% Defining k_comp so we can send it to scope
k_comp = 0;

% Computing k_tau in terms of distance from plane
p_tau =500;
a_tau = -0.002;
k_tau_computed = k_tau/2 * (1+tanh(p_tau*(distance_from_plane-a_tau)));
```

```

% Defining applyVF to see if VF law is active
applyVF = 0;
%----------------------------------------------------------------

% Check if we are getting pose twist
if (max(isnan(pose_twist))~=1) && (enable_motion == 1)
    % We are moving the master if the twist is a valid number (if master is
    % not engaged then twist is NaN). We also check if enable_motion is set
    % to 1

    % Check if we are within the threshold of applying the VF
    if distance_from_plane <= epsilon_plane
        % VF law is active
        applyVF = 1;
        %-------------------------------------------
        % Updating gains to close distances to curve and plane
        if norm(u) < epsilon_pi
            kp_pi = 0;
        end

        if norm(v) < epsilon_c
            kp_c = 0;
        end
        %-------------------------------------------
        % Computing Projection matrices and their null space components
        P_pi = X_pi*pinv(X_pi'*X_pi)*X_pi';
        null_P_pi = eye(3,3) - P_pi;

        P_c = t*pinv(t'*t)*t';
```

```
null_P_c = eye(3,3) - P_c;

twist_in_normal = null_P_pi * pose_twist;

if norm(u) < 0.001
    u_hat = zeros(3,1);
else
    % Getting correction vector back to plane
    u_hat = u/norm(u);
end

if norm(v) < 0.001
    v_hat = zeros(3,1);
else
    % Getting correction vector back to plane
    v_hat = v/norm(v);
end

if norm(twist_in_normal) < 0.001
    twist_in_normal_hat = zeros(3,1);
else
    % Signed normal based on current projection of commanded vel.
    twist_in_normal_hat = twist_in_normal/norm(twist_in_normal);
end
%-------------------------------------------
% Computing universal gain for moving away from plane
if dot(twist_in_normal_hat,plane_normal) > 0
    % Pulling away from plane
    k_tau_computed=k_tau_out_in_ratio*k_tau;
else
```

```
        % Pushing into plane
        k_comp = 0.1;
    end

    %-------------------------------------------
    % Control Equation
    switch VF_mode
        case 1
            % VF on a plane
            % Applying the virtual fixture law (in plane)
            computed_twist = [[ka*((P_pi+k_tau_computed...
                                        *null_P_pi)*pose_twist) + ...
                                    (kp_pi * u_hat)]; twist(4:6,1)];

            % Computing force feedback
            if enable_FF == 1 && distance_from_plane <= 0
                computed_force = K_stiffness * ...
                                            distance_from_plane * u_hat;
            else
                computed_force = zeros(3,1);
            end
        case 2
            % VF on a curve
            % Applying the virtual fixture law (on curve)
            computed_twist = [[ka*((P_c + k_tau*null_P_c)...
                                                    *pose_twist)+ ...
                                    (kp_pi * u_hat) + ...
                                    (kp_c * v_hat)];twist(4:6,1)];

             % Computing force feedback
            if enable_FF == 1 && distance_from_plane <= 0
                computed_force = K_stiffness * ...
                                           distance_from_plane * u_hat;
            else
                computed_force = zeros(3,1);
            end
        case 3
            % VF within a curve
            % Determining gain k_comp in forbidden directions
            p = tanh_gain;
            a = r_curve/scale_r;
            x_from_c = min_distance;
            k_comp = 0.5*(1+tanh(p*(x_from_c - a)));

            % Applying the fixture law
            computed_twist = [[ka*((P_c + k_comp*null_P_c*P_pi + ...
                                          k_tau_computed * null_P_pi)...
                                                      *pose_twist)+ ...
                                   (kp_pi * u_hat) + ...
                                   (kp_c * v_hat)];twist(4:6,1)];

             % Computing force feedback
            if enable_FF == 1 && distance_from_plane <= 0
                computed_force = K_stiffness * ...
                                           distance_from_plane * u_hat;
            else
                computed_force = zeros(3,1);
            end
        otherwise
            % Error mode
```

```
                computed_twist = zeros(6,1);
                computed_force = zeros(3,1);
        end
        %-------------------------------------------
        % Converting computed in omni frame
        computed_force = R_rob2omni * computed_force;

    %-------------------------------------------
    else
        % Condition where we are at some distance > specified_threshold
        % from plane
        applyVF = 0;
        computed_twist = twist;
        computed_force = zeros(3,1);
    end

%-------------------------------------------
else
    % The master is not moving
    computed_twist = zeros(6,1);
    computed_force = zeros(3,1);
end
```

REFERENCES

[1] Brian Davies. The acrobot technology: a model for robotic surgery.

[2] Matjaz Jakopec, Simon J Harris, Ferdinando Rodriguez y Baena, Paula Gomes, and Brian L Davies. The acrobot system for total knee replacement. *Industrial Robot: An International Journal*, 30(1):61–66, 2003.

[3] KH Fuchs. Minimally invasive surgery. *Endoscopy*, 34(02):154–159, 2002.

[4] Paolo Dario, Blake Hannaford, and Arianna Menciassi. Smart surgical tools and augmenting devices. *Robotics and Automation, IEEE Transactions on*, 19(5):782–792, 2003.

[5] Gary Guthart and John Kenneth Salisbury Jr. The intuitivetm telesurgery system: Overview and application. In *ICRA*, pages 618–621, 2000.

[6] William L Bargar, André Bauer, and Martin Börner. Primary and revision total hip replacement using the robodoc (r) system. *Clinical orthopaedics and related research*, 354:82–91, 1998.

[7] Matjaz Jakopec, Ferdinando Rodriguez y Baena, Simon J Harris, Paula Gomes, Justin Cobb, and Brian L Davies. The hands-on orthopaedic robot" acrobot": Early clinical trials of total knee replacement surgery. *Robotics and Automation, IEEE Transactions on*, 19(5):902–911, 2003.

[8] Ming Li, Masaru Ishii, and Russell H Taylor. Spatial motion constraints using virtual fixtures generated by anatomy. *Robotics, IEEE Transactions on*, 23(1):4–19, 2007.

[9] Peter Kazanzides, Tian Xia, Clint Baird, George Jallo, Kathryn Hayes, Nobuyuki Nakajima, and Nobuhiko Hata. A cooperatively-controlled image guided robot system for skull base surgery. *Studies in health technology and informatics*, 132:198–203, 2007.

[10] Jake J Abbott and Allison M Okamura. Virtual fixture architectures for telemanipulation. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 2798–2805. IEEE, 2003.

[11] Alessandro Bettini, Panadda Marayong, Samuel Lang, Allison M Okamura, and Gregory D Hager. Vision-assisted control for manipulation using virtual fixtures. *Robotics, IEEE Transactions on*, 20(6):953–966, 2004.

[12] Ming Li and Russell H Taylor. Spatial motion constraints in medical robot using virtual fixtures generated by anatomy. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2, pages 1270–1275. IEEE, 2004.

[13] Thomas B Sheridan. Human supervisory control of robot systems. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 808–812. IEEE, 1986.

[14] Louis B Rosenberg. Virtual fixtures: Perceptual tools for telerobotic manipulation. In *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*, pages 76–82. IEEE, 1993.

[15] Shinsuk Park, Robert D Howe, and David F Torchiana. Virtual fixtures for robotic cardiac surgery. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2001*, pages 1419–1420. Springer, 2001.

[16] Andreas Hein and Tim C Lueth. Control algorithms for interactive shaping [surgical robots]. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 2025–2030. IEEE, 2001.

[17] Daniel Aarno, Staffan Ekvall, and Danica Kragic. Adaptive virtual fixtures for machine-assisted teleoperation tasks. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1139–1144. IEEE, 2005.

[18] Jacob Rosen, Blake Hannaford, Christina G Richards, and Mika N Sinanan. Markov modeling of minimally invasive surgery based on tool/tissue interaction and force/torque signatures for evaluating surgical skills. *Biomedical Engineering, IEEE Transactions on*, 48(5):579–591, 2001.

[19] Jacob Rosen, Jeffrey D Brown, Lily Chang, Mika N Sinanan, and Blake Hannaford. Generalized approach for modeling minimally invasive surgery as a stochastic process using a discrete markov model. *Biomedical Engineering, IEEE Transactions on*, 53(3):399–413, 2006.

[20] Jake J Abbott, Gregory D Hager, and Allison M Okamura. Steady-hand teleoperation with virtual fixtures. In *Robot and Human Interactive Communication, 2003. Proceedings. ROMAN 2003. The 12th IEEE International Workshop on*, pages 145–151. IEEE, 2003.

[21] Panadda Marayong, Ming Li, Allison M Okamura, and Gregory D Hager. Spatial motion constraints: Theory and demonstrations for robot guidance using virtual fixtures. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1954–1959. IEEE, 2003.

[22] Ming Li and Russell H Taylor. Optimum robot control for 3d virtual fixture in constrained ent surgery. In *Medical Image Computing and Computer-Assisted Intervention-MICCAI 2003*, pages 165–172. Springer, 2003.

[23] Alexander Basilevsky. *Applied matrix algebra in the statistical sciences*. Courier Dover Publications, 2013.

[24] Danica Kragic, Panadda Marayong, Ming Li, Allison M Okamura, and Gregory D Hager. Human-machine collaborative systems for microsurgical applications. *The International Journal of Robotics Research*, 24(9):731–741, 2005.

[25] Mark W. Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot modeling and control*. John Wiley & Sons, Hoboken (N.J.), 2006.

[26] Lorenzo Sciavicco and Bruno Siciliano. *Modelling and control of robot manipulators*. Springer, 2000.

[27] William G. Zikmund. *Exploring marketing research*. Dryden Press, Fort Worth, Tex. [u.a.], 4. ed edition, 1991.