Detection of discrepancies and sensory-based recovery for virtual reality based telemanipulation systems

A. Kheddar†‡, K. Tanie† and P. Coiffet†

† Mechanical Engineering Laboratory
Namiki 1-2, Tsukuba, Ibaraki 305, Japan

‡ Laboratoire de Robotique de Paris
CNRS URA 1778 - UPMC - UVSQ
10-12, Av. de l’Europe, 78140 Vélizy, France

Abstract—Teleoperators using an intermediate functional representation of the remote real environment (RE), suffer a lack of its accurate synthetic modeling. Indeed, discrepancies will always occur between the RE and, let’s say, its artificial representation by means of virtual environment (VE). A strategy to deal with VE/RE uncertainties based mainly on sensory level interpretation is presented. It is directed to avoid use of task knowledge by providing a stream of virtual sensors values. Within the remote site, a supervisor is in charge of recovering from the VE/RE discrepancies by a continuous simulated and real states comparison. The states are derived respectively from simulated and real sensors interpretation. A simple experiment with the proposed approach is presented using only position (velocity) and force sensors. Limitations of the proposed approach are also discussed.

1 Introduction

For various applications, many advanced teleoperators use an intermediate functional representation of the remote RE. For instance, graphic predictors seem to be the more attractive solution to deal with time-delayed remote control and many researchers are applying this approach to real-life tasks like: space robotics, undersea robots, as every strategy, plans, etc. will have to be re-entries. Within the master side, the above cited teleoperators (and many others) are using a VE which displays, via adequate interfaces, real-time synthetic feedback to the operator in response to his/her input commands. These schemes suffer a lack of a realistic RE synthetic modeling. Practically, discrepancies will always occur between the RE and the VE which may affect the teleoperation. These discrepancies might have various origins. In all cases, they can be classified into geometric and dynamic uncertainties. Many designs have been proposed. In [2] the recovery from task errors is somehow a part of the controller because guarded motions (teleprogramming) are sent. In [3] the recovery is also a part of the controller by means of tele-sensory programming 1 where complex tasks are split up into elemental moves for which a certain constraint-frame and sensor type configuration holds (defined by the operator). In [1] the recovery is based on a class of controlled Petri Nets by associating contact states between objects (edge-edge, edge-vertex, plan-edge, plan-vertex, plan-plan) with state places. The solution proposed by [10] is based only on error detection and requires a permanent operator contribution.

The proposed approaches have a common point: at least, a priori task knowledge is needed. A priori task knowledge means that a preprogrammed plan or a strategy to recover from errors is attributed to each task or to a set of tasks/sub-tasks. Unfortunately, for various reasons, they are practically either time-costly or difficult to be applied, especially for hazardous environments and/or applications where tasks are not known in advance 2 . On the another hand, due to the complexity of tasks requirements within a huge amount of probable situations, recovering from VE/RE discrepancies without any task knowledge is actually a challenging proposition.

The motivation of this work derives from the need to improve our precedent experiments [5]. In a more general telerobotic frame, task descriptions require:

— knowing the tasks that have to be performed
— the state of the tasks in the VE is also
— how to describe tasks is also a well known old problem. The adopted description(s) must in general guarantee the determinism of the automata used and prevent state transition by permitting many possible re-entries.

For the third point, it has been stated however, that there is always an abstraction level of task description which is common to both human and robots. Many researchers investigate this way, but still no satisfactory method has demonstrated this common level, especially those using AI techniques. There is also a suitable sensory level of tasks description which may be transparent for both humans and robots. Our aim

1The ROTEX experiment can be considered as persuasive enough about the feasibility and the use of simulated sensors.

2It is globally the privileged areas of teleoperation.
is to find this common denominator exploiting reflex and sensory loops based controllers. A task knowledge independent method, which unburden the VE core is established. A sensory level ‘representation’ is proposed -as it is common and independent from any kind of task- to investigate whether a priori task knowledge can be avoided or not. Theory of this approach and its implementation is presented. Experiments have been conducted to show the feasibility of the proposed method and also to show its limitations. Limitations might be related to the nature of the sensors used.

2 A sensory-level based strategy

There exists in general 2 kinds of discrepancies between the RE and the VE: geometric and dynamic.

Geometric discrepancies may have two sources: (1) geometric errors in the virtual model (non robust reconstitution algorithms, non robust calibration algorithms, etc), and, those ones inherent to the decomposition of continuous surfaces into polyhedra and polygons or complex form simplification to meet both graphical and nominal modeling requirements (2) the second class of geometric discrepancies may derive from the slave robot. More explicitly, from its eventual inability to estimate the exact position of the objects occurring during the grasping, release or assembly phases and also during the manipulation.

Dynamics discrepancies are generally concerned with mass, center of mass, friction coefficient, damping, stiffness, etc. of the environment features involved by the task. They may be partially known or not known at all. This may come from the lack of estimation algorithm, non-linearity of the environment dynamic, etc. It is assumed that robot dynamics parameters are compensated by low level control and uncoupled from the environment ones.

2.1 Notations

Let proceed to the following description:

\( D = (d_1, \ldots, d_n) \) be the vector of the sensory values in a sub-vectors \( D_{\text{snps}} \) of \( D \).

\( S_i = (\sigma_{i,1}, \ldots, \sigma_{i,n}) \) be a binary vector of elementary states. The states derive from condition of \( D \) components or \( S_i \neq S_j \). \( S_i \) design a determined state of the robot and other observable entities. This state is strictly function of internal and external sensors and/or from previously determined states. Let \( N_S = \dim(S_i) \), for each \( S_i \):

\[
N_S = \dim(S_i), \quad i \in [1..N_S]
\]

\( N = (\nu_1, \ldots, \nu_N) \) be the vector of noise values. If \( N_S = \dim(N) \), then \( N_S = N_D \) and

\[
\nu_i = G_i(\delta_i) \quad |\nu_i| \geq 0, \quad i \in [1..N_N]
\]

\( \delta_i \) is a scalar, a vector or a matrix to gather same sensors values in a sub-vectors \( D_{\text{snps}} \) of \( D \).

Each \( N \) component is such that for a measured value \( \delta_{i,m} \) of \( D \). Its real value \( \delta_i \) is considered to be limited:

\[
\delta_i \in [\delta_{i,m} - \nu_i, \delta_{i,m} + \nu_i]
\]

\( T = (t_1, \ldots, t_n) \) be the vector where each component define a tolerated discrepancy margin of a particular defined observable parameter \( t_i \in T \). It can be set in a static or variable mode according to a calculated or heuristically estimated modeling uncertainties. \( N_T = \dim(T) \).

\( C = \{c_1, \ldots, c_n\} \) be the set enumerating low level available control laws. Each low level control law is labeled by a unique component of \( C \). For instance \( c_1 \) may be position control, \( c_2 \) force control, \( c_3 \) impedance control, \( c_4 \) make contact, \( c_5 \) adaptive control, etc. A sub-vector \( D \) constituted with the components of \( D \) may be designed to each control law as a desired command or output. Let \( N_C = \text{card}(C) \).

\( R = \{r_1, \ldots, r_n\} \) be the set of available reading models, where \( N_R = \text{card}(R) \). A reading mode is concerned with the way the supervisor collects the simulated \( D \) stream to recover from an eventual detected error. The supervisor may need for instance to buffer the incoming sensors state from the VE, or skip some, looking for a specific state which may be needed during a recovering strategy.

\( M = (m_1, \ldots, m_e) \) is the vector of a particularly significant previous sensors state. This vector is continuously updated during the telemanipulation. If \( N_M = \dim(M) \) then:

\[
N_M = N_D \quad \text{et} \quad M_t = D_t-t_i,
\]

where \( t \) is the instant teleoperation time and \( t_i \) is the time designed by the supervisor (for instance when a significant change occurs in \( D \), the starting of a recovery strategy, etc.). Here, the term ‘significant’ is linked to the supervision process. In other words, \( M \) is mainly a configuration saving for an eventual backward. Indeed, in the case where a strategy did not succeed to recover a detected error, the robot -under the supervisor command- may come back to the defined \( M \) configuration and perform another strategy if any. Each \( m_i \) is a two field structure. The first field is binary to indicate whether the second field is involved in the saved configuration. The second field is identical to the one defined by \( D \).

The vectors, \( D_t, S_t, N_t, T_t \) can be attributed to simulated (from the VE) or real (from the RE) values according to the item \( x \), then \( x \in \{\text{simu, real}\} \).

Let \( A(a_0, a_0, \ldots, a_0) \) and \( B(b_0, b_0, \ldots, b_0) \) be two vectors such as \( \dim(A) = \dim(B) = N \) then:

\[
A <,>, \leq, \geq B \equiv \forall i \in [1..N] \quad a_i <,>, \leq, \geq b_i
\]

respectively, and

\[
A <,>, \leq, \geq B \equiv \exists i \in [1..N] \quad a_i <,>, \leq, \geq b_i
\]

respectively, and

\[
|A| = (|a_0|, \ldots, |a_n|)
\]

is the absolute vector of \( A \). The maximum, minimum of two vectors is defined:

\[
\max(A,B) = (\max(a_0, b_0) \cdots \max(a_n, b_n))
\]

2.2 Theory of the Proposed Approach

The approach we aim to realize is based essentially on low level sensors data interpretation. Indeed the more efficient is the robot sensors simulation in the
VE the more efficient is the error recovery control in the RE. The teleoperator architecture is shown in Fig. 1. The operator is achieving tasks within the VE which include functional representation of the RE. Direct hand actions are used for teleoperation [4], which means that the real and the virtual robot is hidden to the operator. Necessary bilateral transformations are achieved from the VE to the virtual hidden robot which is also simulated with its sensors. One may use also a direct teleoperation of the virtual robot which is rendered in the VE [7] [2] [3] and skip the bilateral transformation layer. The virtual sensors values are sent in a stream to the slave station in a continuous mode during the teleoperation.

### Figure 1: Global view of the sensory control scheme

In the RE, figure 1, every data (received, actually sensed, to be sent, etc.) transit by the supervisor. According to the actual reading mode $r_j \in R$, the supervisor collects the stream of virtual sensors data coming from the VE. The actual $D_{simu}(t)$ is then constructed from which $S_{simu}(t)$ is derived. A simple state $S_i$ coding is:

$$E_{i, simu/real} = \sum_{j=0}^{N_{x_i} - 1} \sigma_{i,j} 2^{N_{x_i} - j - 1}$$

At the same time the real slave data are collected (no reading mode is specified) after what $D_{real}(t)$ is built from which $E_{real}(t)$ is derived.

At first, $E_{simu}$ and $E_{real}$ are compared. If the two states match then $D_{simu}$ and $D_{real}$ are compared. $D$ is a sub-vector of $D$ such as the components of $D$ are the sensory values involved by the actual state $E_{simu/real}$.

Let $\epsilon = D_{simu} - D_{real}$. A concordance condition is defined by:

$$\epsilon - \overline{E}_{simu} - \overline{E}_{real} \leq \epsilon \leq \epsilon + \overline{E}_{simu} + \overline{E}_{real}$$

or

$$| \epsilon | \leq \overline{E}_{simu} + \overline{E}_{real}$$

Then, the task performed in the VE is assumed to be similarly performed in the RE. Otherwise, the supervisor selects the appropriate control law(s) which may validate this condition. However, if one wants to provide a less restricted condition, we think it is better to shorten the acceptance of VE and RE similarity by using:

$$| \epsilon | \leq \text{max}(\overline{E}_{simu}, \overline{E}_{real})$$

At this stage, it is assumed that if $E_{simu} = E_{real}$ then $\exists C_{\epsilon_{min}} = \in C \mid \text{equation 12 can be achieved}$.

If for the defined states $\exists$ a twosome $(E_{simu}, E_{real})$ such as $E_{simu} \neq E_{real}$, the global strategy is then to choose an appropriate switching of the control laws to make first a state to state conformity in the defined order, i.e. $E_{real} \rightarrow E_{simu}$. The supervisor chooses from $C$ an appropriate sequence to reach first $E_{real} = E_{simu}$, then satisfies equation 12.

During the recovery process, the supervisor runs a tolerance margins checking process. Indeed, the defined tolerance parameters are initialized within $T_{init}$ just before the recovery process. Instant robot configuration is saved within $M$ which is called backwards state (BS). Absolute variations $T_{Z}$ relative to $T_{init}$ are continuously compared with the allowed corresponding ones $t_i$ of $T$ vector, then if:

$$| T_{Z} | > T$$

the recovery process is stopped; the robot is back-driven to $M$. At this stage two issues are possible: 1) another strategy is prepared for, and in this case it is activated, 2) no other strategy exists, the supervisor stops the teleoperation and solicits the strategy from the operator.

For more clarity, without loss of generality, details are given within an implementation instance.

### 3 Implementation and Experimentation

#### 3.1 Description of the Experimental Setup

A 3 dof planar direct-drive robot controlled via a VME bus was used (figure 2). The master station is emulated to be a simple interface where the operator provides on-line a trajectory where the operator provides on-line a trajectory and a state of expected measured forces information. The task is a part mat-

![Figure 2: The experimental setup](image-url)
Many experiments have been performed. Results of the one shown in Fig. 3 are discussed. Within the VE, step $A = 1 \rightarrow 2$ was to move the box towards the fence, step $B = 2 \rightarrow 3$ was mating the box edge to the fence one with a desired force (pure force control), step $C = 3 \rightarrow 4$ was moving the box in free motion, step $D = 4 \rightarrow 5$ moving the box in free motion, step $E = 5 \rightarrow 6$ mating another edge of the fence with another edge of the box, $F = 6 \rightarrow 7$ go back to rest position, then repeat again steps $A, B, C$. Within the VE, this was emulated by a keyboard input of desired $D_{\text{simu}}$ read continuously from a specified file by the supervisor.

Within the RE, the robot was initialized by a false orientation and the fence was disoriented by $\approx \frac{2}{3}$ rad, and translated onto an unknown position according to the robot base frame and within the tolerance constraint.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure3}
\caption{Geometric discrepancies. Experimental VE task, dotted is RE configuration.}
\end{figure}

### 3.2 Implementation of the Recovery Algorithm

As only position and force sensors are used, the operational space parameters of both the virtual and the real robot are used. Let $\bar{\mathbf{D}}_P = (X \ Y \ \psi)$ and $\bar{\mathbf{D}}_F = (F_x \ F_y \ M_t)$ then

$$D_{\text{simu/real}} = (\bar{\mathbf{D}}_P \ \bar{\mathbf{D}}_F)$$

The noise vector can be determined off-line according to sensors resolution and input device (master) noise, virtual force algorithm precision, etc. Then similarly

$$N_{\text{simu/real}} = (\bar{\mathbf{N}}_P \ \bar{\mathbf{N}}_F)$$

In this set up, $N_{\text{simu}} = (0.0 \ \cdots \ 0.0)$ (no noise) and $N_{\text{real}} = (0.02 0.02 0.02 6.0 6.0 5.0)$.

Four possible states: $s = \{s_0 = \text{no motion}, s_1 = \text{free motion}, s_2 = \text{pure force}, s_3 = \text{constraint motion}\}$ are derived from sensor data. For either simulated or real vectors we have

$$s_0 = |\bar{\mathbf{D}}_P| > \bar{\mathbf{N}}_P \ \wedge \ |\bar{\mathbf{D}}_F| < \bar{\mathbf{N}}_F$$

$$s_1 = |\bar{\mathbf{D}}_P| < \bar{\mathbf{N}}_P \ \wedge \ |\bar{\mathbf{D}}_F| < \bar{\mathbf{N}}_F$$

$$s_2 = |\bar{\mathbf{D}}_P| > \bar{\mathbf{N}}_P \ \wedge \ |\bar{\mathbf{D}}_F| > \bar{\mathbf{N}}_F$$

$$s_3 = |\bar{\mathbf{D}}_P| < \bar{\mathbf{N}}_P \ \wedge \ |\bar{\mathbf{D}}_F| > \bar{\mathbf{N}}_F$$

The tolerance vector is defined off-line by the operator. But this is not necessary, if the tolerance notion can be determined dynamically according to some functional features which may improve the recovering procedure. For this experimental purpose, $\overline{T}_P = (10.0 \text{cm} \ 10.0 \text{cm} \ \frac{\text{rad}}{3})$ and $\overline{T}_F = (20.0 \text{N} \ 20.0 \text{N} \ 10.0 \text{N} \text{m}^{-1})$ then

$$T = (\overline{T}_P \ \overline{T}_F)$$

Within the supervisor a tolerance binary flag can either be STOP\(^3\) (set initial values only) or CHECK start to 'integrate' according to the absolute last initial values (simulated and/or real, depending on situations) within $T_S = D_{\text{simu/real}}(t) - D_{\text{simu/real}}(t_{\text{check start}})$ and compare them to the allowed ones defined in $T$.

By a sequential indexing of the VE state, it is possible to restore the VE with the updated data. The index is sent together with $D_{\text{simu}}$. Then it is sent back to the master when an error could not be recovered. When the supervisor solicits the operator, the VE will be restored and updated from this index. Then it is the responsibility of the operator to proceed in a different way.

The reading mode is set as follows,

$$\mathcal{R} = \{r_1 = \text{IGNORE}, r_2 = \text{NEXT}, r_3 = \text{SKIP}\}$$

When the supervisor sets the reading mode to $r_0$, $D_{\text{simu}}$ is ignored and buffered. This has to be seen as a read break used when the supervisor processes a recovering strategy from a detected error. The second usual reading mode $r_1$ enables continuous sequential reading of $D_{\text{simu}}$ whereas $r_2$ mode enables reading and buffering of $D_{\text{simu}}$ under a seeking condition of a particular state. The last one is used also in an on going recovering strategy processing.

A set of low level control laws has been implemented, the number of low level control laws do not improve necessarily the recovering process. It is somehow related to the defined states. In our case we defined

$$c : \{c_0, c_1, c_2 \text{ are reserved commands, } c_3 = \text{POSITION ctrl, } c_4 = \text{ESCAPEp ctrl, } c_5 = \text{FORCE ctrl, } c_6 = \text{make CONTAct, } c_7 = \text{FORCE/POSITION ctrl, } c_8 = \text{ADAPT friction ctrl, } c_9 = \text{COMPliance ctrl}\}$$

This set is not limited to those controllers. One may improve this set by using dynamic control, impedance control, sliding mode control, fuzzy control, neural based control, etc. Some of the control laws are redundant, for instance ESCAPEp control is like FORCE control, but for the supervisor the context of using force control is different. Then, a strategy may be chosen according also to the precedent control law used.

When all those parameters are determined, the supervisor processes on-line as already explained in theory. For each couple of possible states, the supervisor

\(^3\)Which designed also the label.
sets the triplet

\[ < r_i \in \mathcal{R}, c_j \in \mathcal{C}, \text{STOP/CHEC tolerance} > \]  

as follow:

- \( \varepsilon_{\text{simu}} = 0 \)
  - In this case \( \varepsilon_{\text{real}} \) may be in one of 4 possible states:
    - 0: \( < \text{NEXT, POSI, STOP} > \)
    - 1: \( < \text{IGNO, POSI, CHEC} > \)
    - 2: \( < \text{NEXT/SKIP, POSI, CHEC} > \)
    - 3: \( < \text{IGNO, ESCA, CHEC} > \)

- \( \varepsilon_{\text{simu}} = 1 \)
  - In this case \( \varepsilon_{\text{real}} \) may be in one of 4 possible states:
    - 0: \( < \text{NEXT, POSI, STOP} > \)
    - 1: \( < \text{IGNO, POSI, CHEC} > \)
    - 2: \( < \text{NEXT/SKIP, POSI, CHEC} > \)
    - 3: \( < \text{IGNO, ESCA, CHEC} > \)

- \( \varepsilon_{\text{simu}} = 2 \)
  - In this case \( \varepsilon_{\text{real}} \) may be in one of 4 possible states:
    - 0: \( < \text{IGNO, CONT, CHEC} > \)
    - 1: \( < \text{IGNO, CONT, CHEC} > \)
    - 2: \( < \text{NEXT, FORC, STOP} > \)
    - 3: \( < \text{IGNO, FORC, CHEC} > \)

- \( \varepsilon_{\text{simu}} = 3 \)
  - In this case \( \varepsilon_{\text{real}} \) may be in one of 4 possible states:
    - 0: \( < \text{IGNO, CONT, CHEC} > \)
    - 1: \( < \text{IGNO, CONT, CHEC} > \)
    - 2: \( < \text{NEXT, FORC, CHEC} > \)
    - 3: \( < \text{NEXT, FORC, STOP} > \)

It is obvious that a sophisticated implementation is not restricted to this instance and that in general, more than one triplet might be associated to one couple of states. That is to say, if the first triplet fails in recovering from a detected error, rather than asking the operator to intervene, the second one is triggered after coming back to the most significant state defined in \( \mathcal{M} \) associated with the detected error. Then if the second fails, the third one is triggered ...

In this experimental case, during the teleoperation the supervisor saves each successful desired \( \mathcal{D}_{\text{simu}} \) into the \( \mathcal{M} \) vector. Then if an error is detected for the next \( \varepsilon_{\text{simu}} \), the supervisor can backderive the robot to \( \mathcal{M} \) to engage eventually another possible triplet. During the recovery process \( \mathcal{M} \) remains unchanged.

### 3.3 Experimental Results

This section presents experimental curves obtained from a successful recovering of \( [\varepsilon_{\text{simu}}, \varepsilon_{\text{real}}] \) cases stated errors, figure 4. We recommend readers to refer for each step to figures 4, 5, 6. Time unit is taken to be the supervisor sampling period \( \times \approx 100 \text{ ms} \), the robot control loop frequency is 500 Hz.

As seen on the presented figures, step A was performed normally as only velocity (position) control was needed to move the real robot (RR) according to the virtual one (VR). Things were going well for this step until \( t \approx 330T_e \) ms of A to B step transition.

- While in the VE the RR made contact, the RE robot did not. In figures 5 this is shown by \( |D_{\text{Fr,real}}| \leq N_{\text{Fr,real}} \) whereas \( |D_{\text{Fr,simu}}| > N_{\text{Fr,simu}} \). The recovery rule [2,1] was triggered using \( \mathcal{M} \) to determine from the precedent robot state the direction of RR motion looking for an eventual contact and under tolerance checking on.

- The contact was made at \( t \approx 400T_e \) ms. The box/fence interaction was not in the desired force configuration. In figure 5 one may notice the appearance of the moment (Mz), which should not be present, as the needed plane to plane contact involves force on the y direction only. It is shown that it was quickly brought to zero by pure force control to well part the box edge onto the fence face (427 \( \leq t \leq 648T_e \) ms).

- Step (C) was performed by force/position control law. However, due to the fact that the geometric error was not integrated, the robot had to adjust continuously the force and position to fences slope difference. This is shown by the oscillations of the RE Mz and RE Vy figure 5. The oscillation between CM and PF in the RE state (figure 4, \( 650 \leq t \leq 830T_e \) ms), is a consequence of this continuous adjustment.

- The next step (D) was moving the box in free motion. But a problem occur, as near the other part of the fence, the operator did not make contact yet (in the VE), whereas in the RE, the robot did at \( t \approx 1354T_e \) ms. As shown in figure 6, reading mode is switched to SKIP. The data from the VE are skipped (thus not rendered) until \( |D_{\text{Fr,simu}}| > N_{\text{Fr,simu}} \). Then teleoperation is succeeded from this state.

- Step (E) was performed without major problem, the box was slightly adjusted to fit the desired forces by pure force control.

- Next step (F) was performed without any major problem, the RR moved as desired by velocity control until no motion was needed. When the robot was standing at that position, we intentionally applied disturbance force on the gripped objects, around \( t \approx 2000T_e \) ms, as shown in fig. 5. The supervisor rule consists of an escape force control as shown in figure 6 until no force was applied. The remaining steps were performed as the previous ones.

In general, ambiguous cases had to be considered. For instance, when the RR made contact, whereas the VR did not, two strategies are candidate: (1) escape force (2) skip waiting for a contact VE state. It can be solved by choosing at first (2) then if nothing was found in the next \( D_{\text{simu}} \) while the tolerance is ex-
ceed the tolerance check again.

A practical consideration is to let the supervisor run less faster than the low level control loop. In other words, we must let time to low level control laws to be performed before considering the new $D_{tol}$. Not respecting this condition may lead to oscillation of the reading mode and the tolerance check as well as in the states. But it will not corrupt the processing of the supervisor or the recovery.

4 Synthesis

4.1 None recovered errors

The limitation of sensors equipment, may imply some lacks in error recovery process. A class of simple errors, see for instance figure 7, can be stated as not recovered in the RE and the control given back to the operator. Obviously there are many eventuality cases (errors) that can not be recovered, even if they are detectable. It is clear that using only position/velocity and force sensors is not sufficient.

4.2 Undetectable errors

The family of errors as in figure 8, are the most tedious, because when they occur, they can not be detected by the proposed low level strategy. Traditionally, a priori task based knowledge strategies proceed by imposing a set of intermediate steps for the tasks which may induce these kind of errors. This study is aimed to forbid the latter way and it avoids imposing to the operator determining those intermediate steps which are very helpful (at least they can help in detecting the error). A low level strategy must in this
case be generated from the VE, imposing the robot to perform some low level command to ensure the correctness of the final state. For those cases, when using only velocity/force sensors, we do not know how to escape from an a priori task knowledge.

4.3 Bad recovery

There is a family of errors which may not be correctly recovered, see figure 9. In this case the error is detected but not correctly recovered because the supervisor will decide to adjust orientation.

![Figure 8: Case of an undetectable error](image)

![Figure 9: Case of a bad recovering.](image)

Anyhow, if one asks any human being to recover from those errors, he will fail in most of them without vision. For some errors, a task knowledge based strategy may also fail to detect figure 8(right case) and figure 9. At least course robot/RE geometric discrepancies must be recovered first, using another kind of sensors. Then the proposed strategy will recover from small remaining discrepancies and those issued from a gras or a manipulation or dynamic of the RE.

5 Conclusion and Future Work

Not using any kind of a priori task knowledge description to recover from VE/RE discrepancies of virtual reality based teleoperator is a challenging proposition. A sensory-level based strategy that recovers from a class of discrepancies is presented. It has been implemented using a supervisor in a real slave robot teleoperation experiment. The effectiveness of the proposed strategy has been improved to recover from small geometric and dynamics modeling discrepancies. The term small errors is quantified by presenting the tolerance notion which when exceeded the control is fed back to the operator. The limitations of the proposed method demonstrates that unfortunately a wide variety of discrepancies still remain not recovered by using only position/velocity and force sensory information. Some others might not be detected whereas some others might be detected and not correctly recovered. The supervisor at its actual configuration does not exceed 400 lines of instructions (programmed in C language) and does not need any kind of task knowledge. Its design made the implementation very modular and easy to be maintained and updated. The core of the proposed algorithm and thus the software, does not need any conceptual modification when other sensors are added to the robot. On the other hand, if \( N \) are well bounded, the proposed approach does not present actually a false detection of error (i.e. detect an error which is in fact not an error). To improve the robustness of the proposed method more sensors (such as range-sensors) must be added. Future work is focused on proving this proposition and the implementation using a 7 dof robot for more complex tasks.

References


2883