# Robot Companion for Industrial Process Monitoring Based on Virtual Fixtures

Enrico Sita[1], Trygve Thomessen[2], Anthony G. Pipe, Farid Dailami, Matthew Studley

*Abstract*— **In this paper, the use of a monitoring companion is proposed to more efficiently collect the process information during the tuning of industrial systems, and therefore to assist the system integrator in the optimization process for complex robotic installations. The monitoring companion consists of an industrial manipulator (monitoring robot) and a Unity-ROS framework for controlling it. We discuss in this paper the features that allow the solution to be re-used in different industrial application thanks to its compatibility with ROS. In particular, we present the approach based on admittance virtual fixtures and how we use such abstraction to track a moving target (it could be the end-effector of another robot for example). Moreover, the concept of varying compliance is introduced as a way to influence the motion of the monitoring robot on the virtual fixtures in the presence of obstacles. The experiments have been conducted in simulation as well as on real hardware to test the accuracy of the system at respecting the virtual fixtures with both a static and a moving monitoring target, although in the current implementation the varying compliance was not included in the experiments.**

## I. INTRODUCTION

For system integrators, optimizing complex industrial robotic applications (e.g. robotised welding) is a difficult and time-consuming task. This is usually due to discrepancies between the models and the actual behaviour of complex systems, and the system integrator needs to fine tune the final installation by trial and error to obtain the desired quality. This procedure is even more tedious when the operator cannot access the robotic system once in operation and must rely on additional sensors to acquire the necessary process information. However, it is often difficult to find a permanent placement for the sensors to be able to fully monitor the process at any given time during the trials, and this would also be a very expensive and potentially unreliable approach, if applied to all of the robot installations. While it is hard to completely remove this trial and error fashion, it is possible to provide a way to gather process information more effectively that can be used in several robotic installations. It is then proposed to provide the system integrator with a monitoring robot in addition to the robot(s) belonging to the industrial process that needs to be optimized (also referred to as *task robot(s)*). The monitoring robot can be equipped with several different sensors and can be moved into close proximity of any installed robot so that it can be used to collect information from that process during and/or after the operation without interfering. The system operator can control the monitoring robot to change its viewpoint and acquire information from various positions (e.g. inspect a workpiece from different angles). With a more effective way of gathering process data, the system integrator can perform his/her primary task (optimizing the industrial process) more efficiently. Since controlling the monitoring robot is not a primary task, the challenge is to make such interaction as flawless as possible not to overload the operator. The operator will control the monitoring robot with a camera view from its endeffector and via a joystick or similar interface.

The concept and the framework to control the monitoring robot and synchronize it with the task robot has been previously discussed in [1]. This paper instead, focuses on the control strategy to navigate the monitoring robot inside the workspace, which is based on abstract surfaces, called *virtual fixtures* [2], [3] (Conceptual representation in Figure 1).

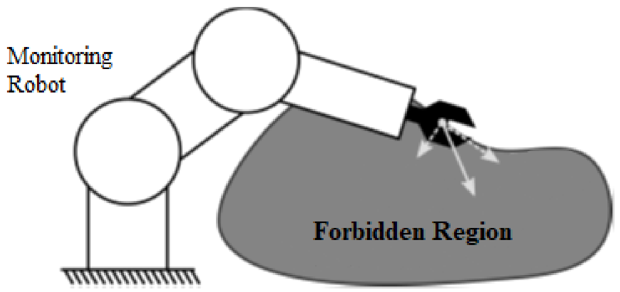


Fig. 1. Conceptual representation of a virtual fixture. The virtual fixture should prevent the robot's end-effector from entering a certain forbidden region.

## II. RELATED WORK

Other researchers have worked on the use of a robot to monitor another industrial process, as it provides flexibility in the choice of viewpoint angle in the workspace as well as allowing any inspection to be performed remotely (so improving EHS conditions when the industrial process is carried on in a harsh environment).

Carvalho *et al.* in [4] discuss virtual reality approaches to be able to inspect an Oil offshore platform, so as to improve understanding during simulation before moving to the real industrial site. Both advanced control techniques and virtual

This work was supported by PPM AS and The Research Council of Norway.

[1]E. Sita, A. G. Pipe, Farid Dailami and Matthew Studley are with the Department of Engineering Design and Mathematics at Bristol Robotics Laboratory, University of the West of England, BS16 1 QY, UK enrico2.sita@live.uwe.ac.uk, tony.pipe@brl.ac.uk, Farid.Dailami@uwe.ac.uk, Matthew2.Studley@uwe.ac.uk

[2]T. Thomessen is with PPM AS, Leirfossvegen 27, 7038 Trondheim, Norway trygve.thomessen@ppm.no

reality representations can improve the situation awareness of the remote site and facilitate the remote control of industrial manipulators that have to act in such remote environment.

In particular, in [5] the authors have proposed an inspection robot to monitor offshore operations on oil and gas platforms. In their paper, Bjerkeng *et al.* presented a flexible camera view based on the weighted pseudoinverse redundancy resolution method. They could autonomously monitor the operation of a second industrial robot while a user could adjust the zoom according to his/her needs. By appropriately dividing the solution space in task space and null space, the monitoring robot could perform its main objective while handling singularities during its motion.

It is worth noticing that an area of improvement could be in extending the adjustment that the user is allowed to make also to the viewing angle (compared to only relative distance adjustments). However, providing more possibilities to adjust the monitoring viewpoint comes with a cost of increasing the difficulty of the overall monitoring robot control. Therefore, this paper illustrates such an approach that allows for more flexibility in the viewpoint control during operation.

## III. VIRTUAL FIXTURES

Virtual fixtures (also called *active constraints*) are a concept introduced by Rosenberg [2] as a way to anisotropically influence robot movements. Active constraints are a very important concept for many telesurgery applications, and have been thoroughly surveyed in this light by Bowyer *et al.* in [6]. For conciseness, the term virtual fixtures will be used instead of active constraints from now on. This work adopts the geometric approach discussed by Marayong *et al.* in [7]. More specifically, the virtual fixtures are represented by a set of *preferred* and *non-preferred* directions of motion which can be designed to be an abstract surface that the robot's end-effector cannot penetrate. The fact that some directions are identified as non-preferred means that the end-effector motion will be less compliant along such directions, as if the end-effector were experiencing some resistance.

It is important to notice that the use of virtual fixtures is independent of the type of control scheme used on the robot, which could be either admittance or impedance control. With a very brief description, we could say that impedance control imposes a force on the robot (spring-mass-damper behaviour), while admittance control imposes a position. With impedance control, forces can be applied to the robot in response to its interaction with the environment. In admittance control, the robot motion is purely decided by the control software and the robot tends to be stiffer when it gets in contact with external objects. In particular contexts, admittance control can be seen as a ”safer” approach compared to impedance control due the absence of motion if the user input drops to zero. However, admittance and impedance control could be interchanged in many applications. This paper focuses solely on admittance control, meaning that the control software will filter the user input and apply the filtered motion to the master reference (which in this case is a 3D model of the monitoring robot).

The way to achieve this behaviour in admittance control, is to filter the user input commands along the directions described by the virtual fixture.

More particularly, let us assume then to have a $6 \times n$ matrix $D = D(t)$, $1 < n < 6$ containing the preferred directions of motion. The dimension of the $D$ matrix is determined by the type of constraint imposed on the end effector. For example, if $n$ is 1, the preferred motion is along a curve in $SE(3)$, or along a surface if $n$ is 2 and so on. As described by Marayong *et al.* in [7] and by Hager in [8] the input vector can then be decomposed along preferred and non-preferred directions with the Kernel and Span operators:

$$v_D \equiv [D]\mathbf{f}_{in} \quad \text{and} \quad \boldsymbol{v}_\tau \equiv \langle D \rangle \mathbf{f}_{in} \tag{1}$$

where $\mathbf{f}_{in}$ is the vector containing the user input motion.

Due to the properties of the Kernel and Span operator (for more details see Hager in [8]), and since it is possible to write $v_D + v_\tau = \boldsymbol{v}$, it is possible to write the following relationship:

$$\boldsymbol{v} = c([D] + c_\tau \langle D \rangle)\mathbf{f}_{in} \tag{2}$$

where $c_\tau \in [0,1]$ is the compliance factor for the non-preferred directions. The smaller the value of $c_\tau$, the smaller the compliance along the non-preferred directions of motion. If $c_\tau$ is chosen to be equal to zero, the virtual constraint is a *hard* virtual fixture, as opposed to any other value which instead would still permit motion along the non preferred directions.

It is worth observing that, with such definition, the monitoring robot's end-effector can move on a path that is *parallel* to the preferred directions of motion specified in $D$ at every given time, and this is typically the case if, at every instant, the robot moves along the tangent plane (or line) of the abstract surface. In fact, the linearisation error accumulates over time and that results in the end-effector moving onto a parallel path. Instead of modifying the $D$ matrix by including an ”attraction” term, the linearisation error is automatically corrected after every user input command, in order to eliminate the *drifting* component that tends to move the end-effector away from the virtual fixture. If this correction is performed at every iteration, provided that the user is controlling the robot, the ”active” movement that is imposed on the robot at each iteration is small enough not to be a concern in terms of safety.

## IV. LOCAL VIRTUAL FIXTURES AND AUTONOMOUS TRACKING

An additional observation is that such virtual fixtures are not *static* in the 3D environment where we control the monitoring robot. Instead, the virtual fixtures can move, typically together with the end-effector of the task robot. In the literature they are called *dynamic virtual fixtures* or *dynamic active constraints* [6].

However, we believe the use of this term might be inappropriate and misleading here because in this paper we have designed a virtual fixture that in certain situations can be manipulated by the user to influence the navigation capabilities of the monitoring robot.

This section describes the use of a non-compliant virtual fixture that can be manipulated by certain user commands. This virtual fixture can be used together with autonomous tracking motions also when a user performs manual adjustments. Figure 2 shows the conceptual representation of the local virtual fixture (LVF) with the shape of a sphere. The end effector of the monitoring robot is constrained on the sphere's surface and the whole fixture can move in space according to the motion of the task robot's end-effector.

The virtual fixture is initialized with respect to a point or object in the 3D environment, such as the task robot tool tip or for example the workpiece. If the point to which the local virtual fixture is anchored moves inside the workspace, then the whole fixture moves along as well. As a consequence of this motion, the monitoring robot is also moving in order to remain on the virtual sphere's surface. This behaviour is the motion that is generated by the autonomous tracking part. The monitoring robot motion which is instead generated by user input is only affecting the position of the monitoring robot's end-effector relatively to the virtual fixture.

Figure 3 shows schematically how the monitoring robot's reference is influenced by both user motion and autonomous tracking motion. The monitoring robot can be controlled by any combination of these two sources.

Another property of the local virtual fixture is that in certain scenarios the user can manually modify the anchor's position in the workspace. This scenario is contemplated because the user should be able to change the "focus point" during the operation, meaning that the monitoring robot can be set to inspect different areas of the workspace while being constrained to the surface of the local virtual fixture.

$$\boldsymbol{v} = \boldsymbol{v}_u + \boldsymbol{v}_a \quad \text{and} \quad \boldsymbol{v}_u = c([D] + c_\tau \langle D \rangle)\boldsymbol{v}_{in} \tag{3}$$

where $\boldsymbol{v}_a$ is the velocity vector of the autonomous navigation part. In (3) $\boldsymbol{v}_u$ is the user velocity vector that in (2) was simply called $\boldsymbol{v}$.

## V. VARYING COMPLIANCE

Depending on the application scenario, a noncompliant virtual fixture can bring advantages as well as disadvantages. In this particular case, the zero compliance property of the local virtual fixture conveniently allows to fulfil the task implicit objectives. These objectives are the orientation of the monitoring robot end-effector (i.e. the camera view) towards the anchor point (the object that is being inspected) and a minimum distance from the workpiece or task robot for example.

However, the local virtual fixture does not give any constraint about how the motion should be influenced when approaching other obstacles. Currently obstacles (or forbidden region that can be regarded as obstacles) are expressed in the 3D environment as virtual fixtures so to prevent their contact with the monitoring robot. In the current implementation, however, it possible that the local virtual fixture partially intersects one or more obstacles' forbidden regions if the user decides to change the inspection target by moving the local virtual fixture inside the workspace.

Whenever this happens, it is possible to provide more than just haptic feedback when the monitoring robot comes in contact with the obstacles virtual fixtures (as it moves along the LVF surface). The compliance of the system can be modified to simulate an increase in friction as the robot approaches an obstacle, even though the robot still only moves along the *preferred* directions of motion.

It is important to notice that in the case of a partly compliant VF, a similar result could be achieved by gradually changing $D = D(t)$ to exclude the direction of motion that would move the robot toward the obstacle.

Let us assume to have a spherical LVF, that intersects a certain obstacle represented by its own VF, $S_{obst} \in SE(3)$. Let us continue by assuming that the set of point of the intersection is known, and that for any position of the end effector $\boldsymbol{x}_{tcp}$ it is possible to determine the point $\boldsymbol{P} \in S_{obst}$, and belonging to the intersection, that is closest to the monitoring robot's end effector. Let us introduce the varying compliance $c = c(\boldsymbol{x}_{tcp}(t))$, function of the end-effector position $\boldsymbol{x}(t)$ :

$$c(\boldsymbol{x}(t)) = \begin{cases} 1 & dist(\boldsymbol{x}_{tcp}, \boldsymbol{P}) > h, \quad 0 < h \leq R_{LVF} \\ w_s dist(\boldsymbol{x}_{tcp}, \boldsymbol{P}) & dist(\boldsymbol{x}_{tcp}, \boldsymbol{P}) \leq h, \quad 0 < w_s \leq 1 \\ 0 & dist(\boldsymbol{x}_{tcp}, \boldsymbol{P}) = 0 \end{cases} \tag{4}$$

where $h$ is a threshold that determines at what distance the system starts having less compliance on the preferred directions, while the term $w_s$ is a scaling factor that can be conveniently chosen as $w_s = \frac{1}{h}$ so that $c(\boldsymbol{x}(t)) \in [0,1]$. If the LVF is spherical, $dist(\boldsymbol{x}_{tcp}, \boldsymbol{P})$ corresponds to the spherical distance. If the LVF is not of canonical shape (e.g. is a sensor generated 3D surface) the distance between two points is computed as the distance between two nodes on a graph, where the graph is obtained by sub-sampling the 3D surface.

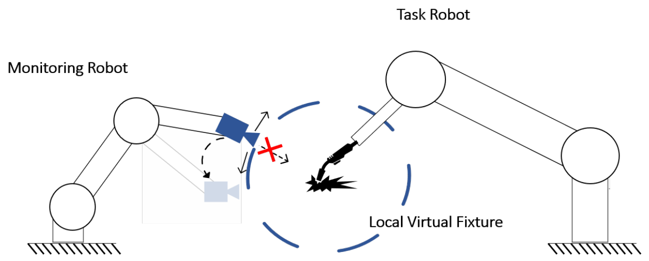


Fig. 2. A local virtual fixture (spherical surface) to monitor an external process.

## VI. SPHERICAL VIRTUAL FIXTURE IMPLEMENTATION

A local virtual fixture with spherical shape has been implemented (see figure 2), to test the behaviour of the monitoring robot during motion.

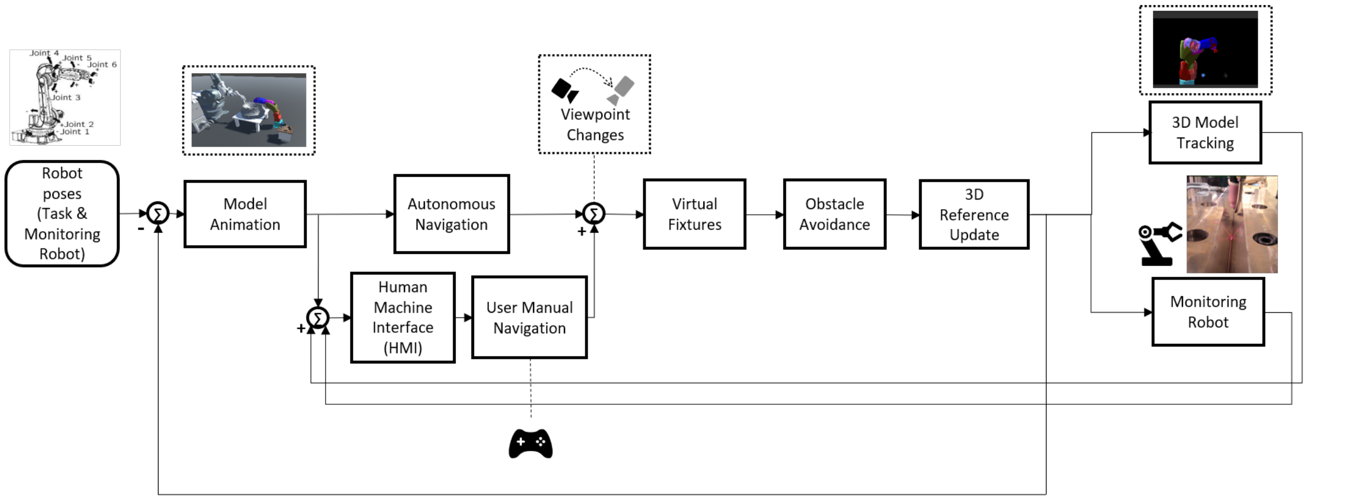


Fig. 3. The simplified control diagram of the monitoring software. Autonomous navigation (or motion) input are in parallel with the user manual navigation. The user can manually adjust the viewpoint of the monitoring robot during the operation, and the changes issued are added to the autonomous tracking motion.

The form of the $D$ matrix then has to describe the preferred motions as the sphere tangent plane that passes through the monitoring robot's end-effector. During the initialization, the control software makes sure that the monitoring robot's end effector is touching the virtual fixture surface, by changing it's radius if necessary. The $D = D(t)$ matrix takes the form:

$$\mathbf{D} = \begin{bmatrix} t_{x_1} & t_{y_1} & t_{z_1} & 1 & 0 & 1 \\ t_{x_2} & t_{y_2} & t_{z_2} & 1 & 0 & 1 \end{bmatrix}^{\mathbf{T}} \quad (5)$$

where $(t_{x_1}, t_{y_1}, t_{z_1})$ and $(t_{x_2}, t_{y_2}, t_{z_2})$ are the vectors that identify the tangent plane to the sphere's surface at any given time. The tangent vectors will change as the end-effector moves along the sphere's surface, making the $D$ matrix time dependent as expected. The elements of $D$ equal to one indicate the freedom to rotate along the $Z$ and $X$ axis in order to maintain the focus toward the centre of the virtual fixture.

The framework to control the monitoring robot is based on Unity3D [9], where a 3D replica of the monitoring robot serves as the reference for the real hardware (the real robot is shown in figure 4). The pose of both the 3D reference and the real robot were recorded to test the performance of the control software during certain test trajectories. In particular, the monitoring robot motion has been observed when performing *ideal* manual movements, when autonomously tracking a moving point, and in a combination of the two motions.

Manual movements are *ideal* because they are actually simulated via software for reproducibility purposes. The input commands are simulated as coming from the same interface that would be used by the human operator (a Joystick) but without noise in the resultant motion.

## A. Manual Motion

The trajectory used for the manual motion is composed of four simple steps:

1. Move "left" with respect to the monitoring tool frame (or camera view) while zooming *out*
2. Move "right" with respect to the monitoring tool frame (or camera view) while zooming *in*
3. Move "right" with respect to the monitoring tool frame (or camera view) while zooming *out*
4. Move "left" with respect to the monitoring tool frame (or camera view) while zooming *in*

The robot starting position, the zooming speed and movement speed are reported in table I.

The resultant path has been performed ten times, shifting the monitoring robot's end-effector "downward" (w.r.t. the monitoring tool frame) of 0.5 millimetres between each

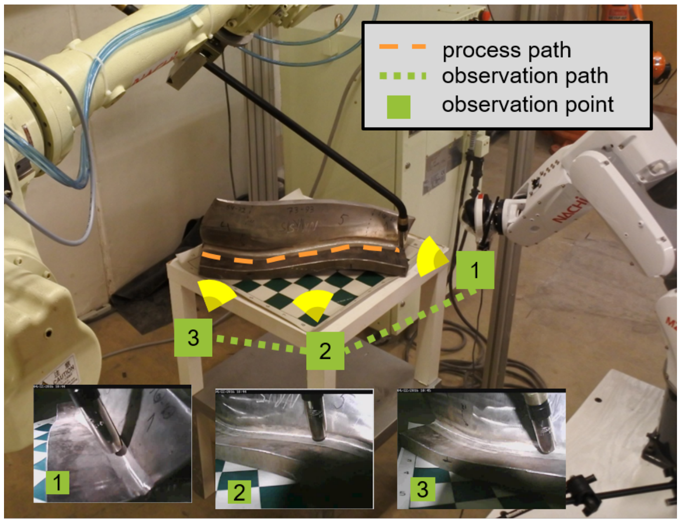


Fig. 4. The industrial setup where a monitoring robot (on the right) can inspect a workpiece. The way-points shown in the picture are selected via user adjustments as the process is being carried out. Note that the relative positioning between the monitoring robot and the task robot changes from one point to the other.

| | |
|---|---|
| **Starting Position ($J_1$,$J_2$...$J_6$)** | (0.002, 89.998, -0.004, 0.005, -89.997, -2.324) [deg.] |
| **Starting Position (x,y,z)** | (0.280, 0.553, 0) [m] |
| **Zooming Speed** | 1.0 [cm/s] |
| **Movement Speed** | 1.0 [cm/s] |
| **Initial LVF Radius** | 10 [cm] |
| **Maximum LVF radius** | 15 [cm] |

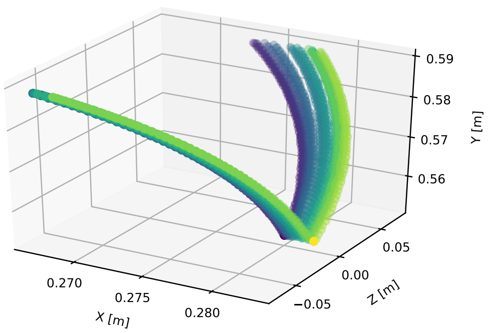
TABLE I
PARAMETERS FOR THE MANUAL MOTION TEST.

iteration, as can be noticed by the drift along the X-axis in figure 5.

Moreover, in this test the user input not only modifies the position of the monitoring robot on the LVF, but also directly modifies the radius of the virtual sphere (the zooming motion). Although changing the radius of the LVF might seem to contradict the original purpose of a virtual fixture, such degree of freedom is normally not enabled and can only be used by a human operator. Finally, the spherical virtual fixture radius has a minimum value of operation that not even the user in manual motion can override. This radius limit prevents the robot from getting too close to the monitoring target. The tracking error between the 3D model and the real robot while performing the manual motion test is shown in figure 6.

### B. *Manual & Autonomous Motion*

The other round of tests consists into combining motion commands from the joystick interface with the autonomous



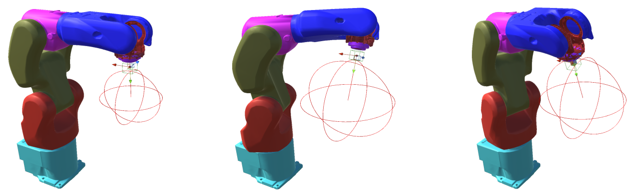


Fig. 5. Trajectory for the test with manual motion. The robot moves along the surface of the local virtual fixture, which has a fixed position.

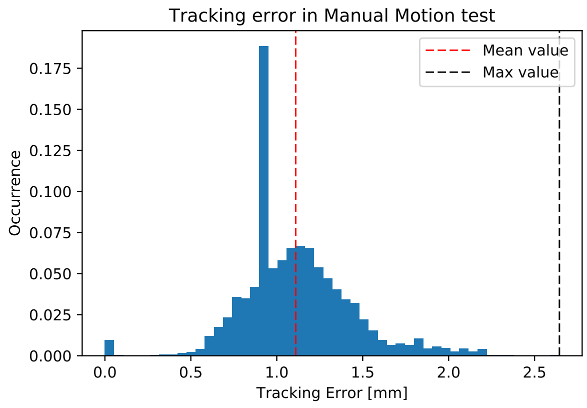
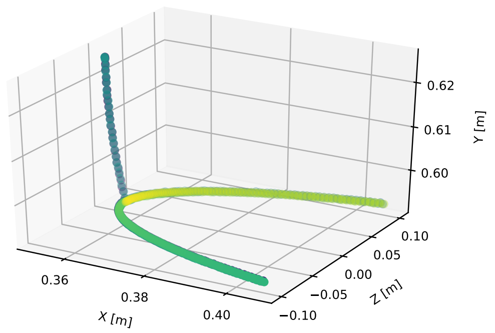


Fig. 6. The histogram of the tracking error during the manual motion test.



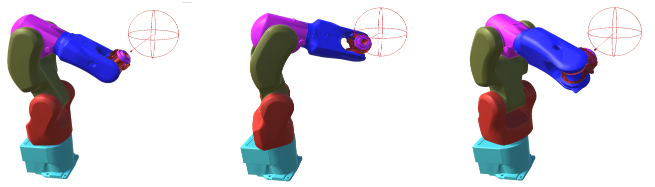


Fig. 7. Trajectory used for the manual & autonomous test. In this picture only the manual motion is displayed.

tracking motion. More specifically, in the Unity environment the local virtual fixture is initialized anchored to a point in space (referred to as ”tracked point” in table II) that will start moving back and forth along a line parallel to the Z-axis, defined by two predefined positions (see table II).

The trajectory generated by manual input instead consists into a downward movement (vertical axis) followed by a repetitive left/right movement along the X axis of the end-effector frame, always constrained to be on the virtual sphere's surface. The trajectory followed by the robot if there was no input coming from the autonomous tracking software ($\boldsymbol{v}_a = 0$) is shown in figure 7.

The path that results from the combination of the two motion commands is instead shown in figure 8, and the parameters for the manual and autonomous motion test are reported in table II.

| | |
|---|---|
| **Robot Starting Position ($J_1$,$J_2$...$J_6$)** | (0.002, 89.998, -0.004, 0.005, 0.002, -0.004) [deg.] |
| **Robot Starting Position (x,y,z)** | (0.352, 0.625, 0) [m] |
| **Tracked Point: Initial Position** | (3.023, -0.375, 0.15) [m] |
| **Tracked Point: Final Position** | (3.023, -0.375, -0.15) [m] |
| **Robot Zooming Speed** | 1.0 [cm/s] |
| **Robot Movement Speed** | 1.0 [cm/s] |
| **LVF Radius (constant)** | 10 [cm] |
| **Tracked Point Speed** | 2 [cm/s] |

TABLE II

PARAMETERS FOR THE MANUAL AND AUTONOMOUS MOTION TEST. THE LVF RADIUS REMAINS CONSTANT DURING THE TEST. THE TRACKED POINT MOVES LINEARLY BETWEEN THE INITIAL AND THE FINAL POSITION.

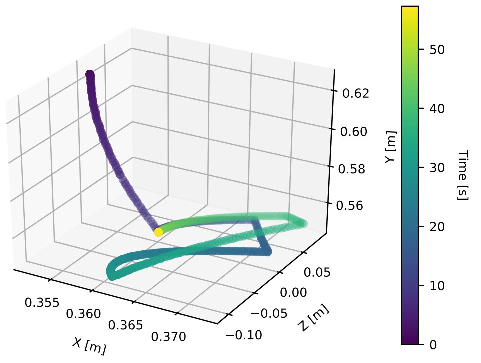


Fig. 8. Path generated by the combination of autonomous tracking movement and manual input commands.

However, it is important to observe that the autonomous tracking motion can generate complications during operation if the monitoring task is not designed appropriately. The monitoring robot can track an abstract point that is only moving in the 3D representation of the scene (as in the current experiment) or, alternatively, it can track the task robot's end-effector. In any case, the autonomous component of the motion can set the monitoring robot after unreachable poses (typically because task robot and monitoring robot have different sizes). Currently, the autonomous tracking is disabled whenever this problem arises, and the LVF anchored to the last reachable position, in order to give priority to the local virtual fixture constraints that still have to be fulfilled by the robot. This possibility is a known problem that will be kept under observation also in future experiments.

## VII. CONCLUSIONS & FUTURE WORK

This paper discussed how our monitoring robot is capable of moving inside the workspace respecting the constraints imposed by the local virtual fixture. It is then possible to inspect a certain workpiece from different angles while respecting constraints like ”look at” orientation and minimum distance from the objective. With this abstraction, the monitoring robot can still perform the inspection on a moving target as shown in the experiments, and manual user adjustments are still permitted during the operation.

Moreover, the concept of varying compliance has been introduced as an approach to regulate the monitoring robot motion on the virtual fixture in the presence of obstacles or other critical forbidden regions.

However additional complications can occur as the monitoring robot moves in the workspace. It might happen that the monitoring robot cannot reach a certain viewpoint and remains stuck due to reachability limitations. Moreover, it is important to evaluate how the use of local virtual fixtures and varying compliance are perceived by the user.

Usability is a very important factor, since the user should be able to operate the monitoring robot for workspace inspection without a sensible increase in workload.

## ACKNOWLEDGMENT

This research is supported by The Research Council of Norway through the Industrial PhD project 264145/O30 Interactive robot operation using multi-modal man-machine communication”

## REFERENCES

[1] E. Sita, C. M. Horváth, T. Thomessen, P. Korondi, and A. G. Pipe, “Rosunity3d based system for monitoring of an industrial robotic process,” in *System Integration (SII), 2017 IEEE/SICE International Symposium on*. IEEE, 2017, pp. 1047–1052.

[2] L. B. Rosenberg, “The use of virtual fixtures as perceptual overlays to enhance operator performance in remote environments.” Stanford Univ Ca Center for Design Research, Tech. Rep., 1992.

[3] ——, “Virtual fixtures: Perceptual tools for telerobotic manipulation,” in *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*. IEEE, 1993, pp. 76–82.

[4] F. Carvalho, A. Raposo, I. Santos, and M. Galassi, “Virtual reality techniques for planning the offshore robotizing,” in *Industrial Informatics (INDIN), 2014 12th IEEE International Conference on*. IEEE, 2014, pp. 353–358.

[5] M. Bjerkeng, A. A. Transeth, K. Y. Pettersen, E. Kyrkjeb, and S. A. Fjerdingen, “Active camera control with obstacle avoidance for remote operations with industrial manipulators: Implementation and experimental results,” Sept 2011, pp. 247–254.

[6] S. A. Bowyer, B. L. Davies, and F. R. y Baena, “Active constraints/virtual fixtures: A survey,” *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 138–157, 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6634270

[7] P. Marayong, M. Li, A. M. Okamura, and G. D. Hager, “Spatial motion constraints: theory and demonstrations for robot guidance using virtual fixtures,” vol. 2. IEEE, 2003, p. 1959 vol.2. [Online]. Available: http://ieeexplore.ieee.org/document/1241880

[8] G. D. Hager, “Human-machine cooperative manipulation with vision-based motion constraints,” in *WS2: Workshop on Visual Servoing*, 2002, p. 1.

[9] Unity3d.com. (2018) Unity 3d: Game engine. https://unity3d.com/ and https://unity3d.com/public-relations. [Online]. Available: https://unity3d.com/