Safe collaboration between human and robot in a context of intermittent haptique interface

Stanley Mugisha, Matteo Zoppi, Rezia Molfino, Vamsi Guda, Christine Chevallereau, Damien Chablat

SAFE COLLABORATION BETWEEN HUMAN AND ROBOT IN A CONTEXT OF INTERMITTENT HAPTIQUE INTERFACE

Stanley Mugisha
Matteo Zoppi
Rezia Molfino
PMAR Robotics Group, DIMEC
University of Genova,
Via Opera pia 15/A, Genova, Italy
Emails: Stanley.Mugisha@edu.unige.it
Matteo.Zoppi@unige.it
Rezia.Molfino@unige.it

Vamsi Guda
Christine Chevallereau
Damien Chablat*
Laboratoire des Sciences du Numérique de Nantes
UMR CNRS 6004,
1 rue de la Noë, 44321 Nantes
Emails: Vamsikrishna.Guda@ls2n.fr
Damien.chablat@cnrs.fr
Christine.Chevallereau@ls2n.fr

ABSTRACT

In the list of interfaces used to make virtual reality, haptic interfaces allow users to touch a virtual world with their hands. Traditionally, the user's hand touches the end effector of a robotic arm. When there is no contact, the robotic arm is passive; when there is contact, the arm suppresses mobility to the user's hand in certain directions. Unfortunately, the passive mode is never completely seamless to the user. Haptic interfaces with intermittent contacts are interfaces using industrial robots that move towards the user when contact needs to be made. As the user is immersed via a virtual reality Head Mounted Display (HMD), he cannot perceive the danger of a collision when he changes his area of interest in the virtual environment. The objective of this article is to describe movement strategies for the robot to be as fast as possible on the contact zone while guaranteeing safety. This work uses the concept of predicting the position of the user through his gaze direction and the position of his dominant hand (the one touching the object). A motion generation algorithm is proposed and then applied to a UR5 robot with an HTC vive tracker system for an industrial application involving the analysis of materials in the interior of a car.

*Address all correspondence to this author.

1 Introduction

The aim of virtual reality is to immerse a human being in a virtual environment using all his senses. In most collaborative systems, the main senses are sight, then hearing, and finally, touch [8]. The sense of vision can be rendered by using large screens that occupy the user's entire field of vision or by using a Head Mounted Display (HMD). In the latter case, the user's vision becomes completely disconnected from the real world and all his movements can become dangerous. In some cases, user may lose his spatial landmarks and have the feeling of falling on the ground. Sound immersion further increases this immersion and separation from the real world. By using immersion HMD and headphone, the user can free himself from his environment.

Haptic interfaces, such as Virtuose 6DOF [23], are used in product design by engineers [18]. In [12, 5] a five-fingered haptic interface robot with a 6 degree of freedom arm and a 15 DOF hand was used to provide multipoint contact between the user and a virtual environment through force and tactile feeling to the fingertips of the human hand. These interfaces are safe and well mastered but if the user can apply force/torque, he cannot really feel the textures and appreciate the quality of the materials.

Among the main shortcomings of these interfaces are limited workspace, low stiffness and high cost.

New haptic interfaces using an industrial robot or a cobot (robots specially designed to work in human-robot environments) can be used as haptic interfaces with intermittent con-

tacts [2, 13]. For the application envisaged in this document, the cobot carries several texture specimens on its end-effector, to allow contact between a user's finger and the robot. They are called Intermittent Contact Interfaces (ICI) [15].

When the user uses HMD vision interfaces and has to perform haptic evaluations, he no longer sees the real scene, but only a virtual world. His physical reference points quickly disappear except for objects he touches such as his seat and the floor.

When users reach to grasp objects, they look at the target first, then bring the hand to the center of gaze to grasp the object. Eye-hand coordination is a fundamental behavior that humans use to interact with the world [10, 11, 20]. The head movement facilitates subsequent gaze shifts toward the future position of the hand to guide object manipulations, thus leading to a strong correlation between head and hand movement parameters [21, 26, 27].

Through the user's gaze and hand movements, as well as the position of areas to be studied, it is possible to predict the tasks that the user will perform. The purpose of this study is to ensure that the robot end-effector will be available for intermittent contact in complete safety when the human hand is close to the surface to touch.

The outline of this article is as follows. First, we present the context of the study and the material used. Then, an algorithm for predicting the user's intention through his gaze and the position of his dominant hand (the one touching the object) is described and implemented under the Unity software. Finally, a motion generation algorithm is proposed and then applied to a UR5 robot with an HTC vive tracker system for an industrial application involving the analysis of materials in the interior of a car.

2 Description of the Context

The context of the study is the evaluation of the perceived quality of a virtual car interior during the first design phases. In a given scenario, the user sits in the real world for a visual virtual reality experience inside the car. The user wears a HMD and cannot see the robot, which explains the safety problem (Figure 1). While the user is trying to interact with the virtual object of the environment, the robot must come and position a sample of the material associated with the local surface, to provide a tactical sense of touching the object [19, 25]. A motion capture system based on HTC vive trackers is used to know the position of the body and especially the hand used for interaction as well as the position of the chair and the robot [28] (Figure 2). Currently, the prop can carry six different materials. The robot is fixed on a 75 cm high table and the user sits on a seat 60 cm above the floor. The placement of the robot in the scene has been chosen to be able to reach all the places where the user's hand will want to have haptic interaction with the robot's probe[9].

A virtual model under the Unity 18.4 LTS software represents the fixed objects in the environment, as well as the moving

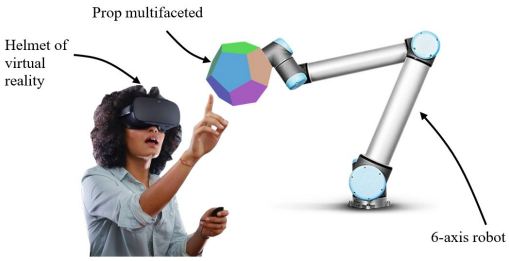

FIGURE 1. Conceptual scheme of the experimental platform

FIGURE 2. The complete system setup for human robot interaction

objects, which are the robot thanks to the encoders of the motors and the user thanks to HTC trackers located on the hands and on its seat. The industry partner provided the virtual model of the car design.

An industrial robot can perform powerful and fast movements that can be dangerous for the humans around it. Involuntary contact between the robot and humans is a threat. This is particularly important in a virtual reality context where humans equipped with an HMD will not be able to anticipate the robot's movements. Today more than ever, men work closely with robots. In the case of intermittent contact interface ICI, contact is inevitable between humans and robots. Cobots are best suited to such a scenario, but in terms of human safety, accident prevention can always be improved [4]. These robots are designed to work at limited speeds during potential contacts. Moreover, it must be ensured that the desired contact with the robot during interaction will not result in a necessary restart of the robot after a safety stop [16].

Modulation of the robot's speed according to the robot's location in relation to man is now our main objective.

3 Human Intention Prediction

3.1 Sensors used

To ensure the safety of the user it is necessary to model the user in the virtual world. Depth cameras allow a faithful reconstruction of the environment but their acquisition frequency is

low with latency. Moreover, camera placement is difficult because the robot can hide the user.

The Figure 3 shows all the sensors used to reconstruct the position of the user's hands and arms with HTC vive trackers using a mannequin model. Two vive trackers are used to reconstruct a single arm. The joints of the arm are selected similar to software like Delmia. For this application, only the arms and the trunk can move because the user remains seated in his chair. This point is not developed further in the paper.

The HTC vive tracker placed on the hand of the human are also used to control the robot end effector. The hand motion defined the the desired location of the end effector as shown in Fig. 4.

3.2 Detection of the Target of Human Motion

Robots need to anticipate human's future actions and act accordingly while performing collaborative tasks. In most human-robot collaboration systems, the motion of robots is based on some predefined programs, which are task-based. However, most tasks are highly complex and it is difficult to redefine a complete set of instructions for such situations. In such tasks, the roles of the robot should be changed from purely automated machines to autonomous companions. Previous works relied on supervised learning methods to build models of human motion, which relied on understanding the environment, offline training or manual labeling, adaptation to new people, and motion styles.

An expectation-maximization (E-M) algorithm and a neural network to infer human intentions in a 3-dimensional (3D) space were used in [24]. They modeled a function with intentions as parameters and developed a neural network to learn human arm dynamics. In [22] time series analysis for the motion of the human arm based on demonstrations of human arm reaching motion, which synthesized anticipatory knowledge of human motions and subsequent action steps to predict was used. A combination of a two-layer framework of Gaussian mixture models and unsupervised learning to predict a remainder of the trajectory from a prior observed human arm motion in reaching tasks was used in [17]. In [14] a Markov decision process to anticipate a belief about possible future human actions was used by modeling the human's and robot's behavior and then constructed a graph to represent the human motion and interaction with objects.

Human intention is mainly expressed through the behavior of humans and the objects they interact with. Most of the current research on human intention prediction just focuses on action classification, in which the human action is classified into several categories, such as running, walking, jumping [6] which is inadequate for accurate inference of human intention in human-robot collaboration.

We propose an HRI framework that combines hand motion with gaze direction to build models on the fly which predict human intention in virtual reality and move the robot to the required position in a virtual space without offline training.

3.3 Proposed Model

We proposed a k-d tree-based model that defines the relationship between human hand and scene interaction taking into account the gaze direction of the user. The main advantage of this approach is that it can build models on the fly and with no offline training. We further examine the limitations of the plain k-d tree based only on the hand and scene objects interaction and the how gaze direction improves the predictions and performance.

3.3.1 Scene Information

From the FBX model of the car in unity3D Virtual reality software, we defined the Regions of Interest (ROI) the user is to interact with. Each ROI is represented as a cube paced at the center of the surface. We have defined 17 regions to be studied inside the car (Figure 5). They are located as follows:

- 4 point on the door,
- 4 points on the chair,
- 4 points on the dash board,
- 1 point on steering wheel,
- 1 point on touch pad,
- 2 points on glove compartment,
- 1 point on speedometer.

3.3.2 Robot Motion

To move the robot to a specific position and orientation, the position and orientation of the cube is sent through a UDP connection to a robot controller at a frequency of 50 Hz. This controller runs on a thread library that consists of two threads, which read data from Unity VR software and write data to the real-time data interface of the UR5 at a frequency of 100 Hz.

3.4 Methodology

We used a proximity search, which is an optimization problem of finding a point in a closed set that is closest to a given point. Closeness is defined by a dissimilarity function such that the dissimilar the objects, the larger the function values.

3.4.1 Problem Definition

Given a set P of our interest points in a 3D space D, and a query point q which represents the user's dominant hand, we find the closest point in P to q. This problem can be generalized as a k-nearest neighbor (kNN) query where we have to find the k closest points where k $\in Z^+$. Implementing the kNN is normally done by computing the distances from q to all elements in P. However, this method is computationally intensive for a large number of data points. We used a k-d tree for the nearest neighbor search proposed by [3, 7]

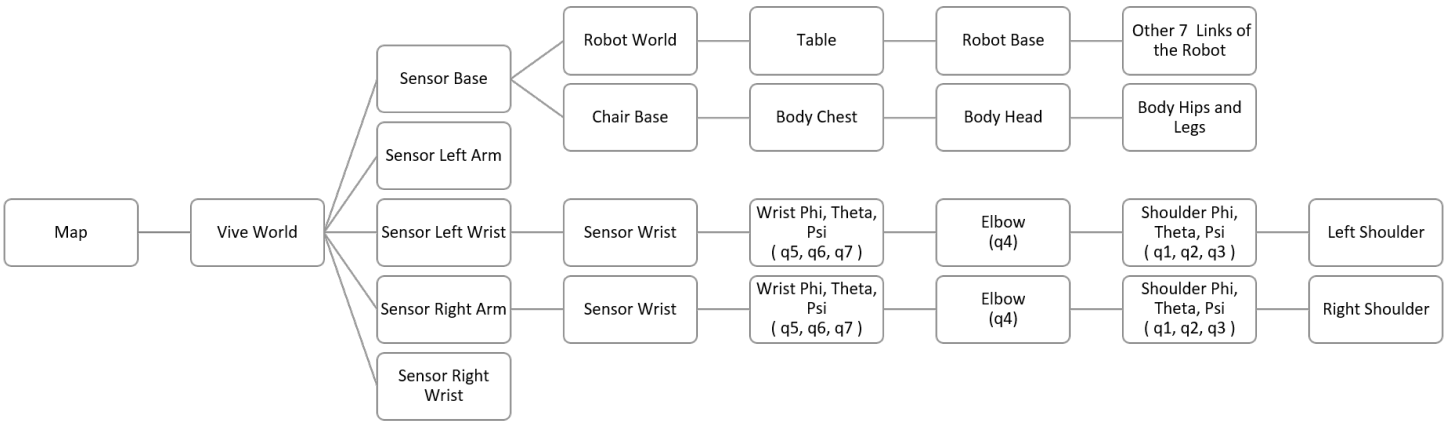

FIGURE 3. Localization of the HTC vive trackers to reconstruct static and dynamic objects in the environment

FIGURE 4. Human with HMD and hand tracker to control the movement of a UR 5 robot

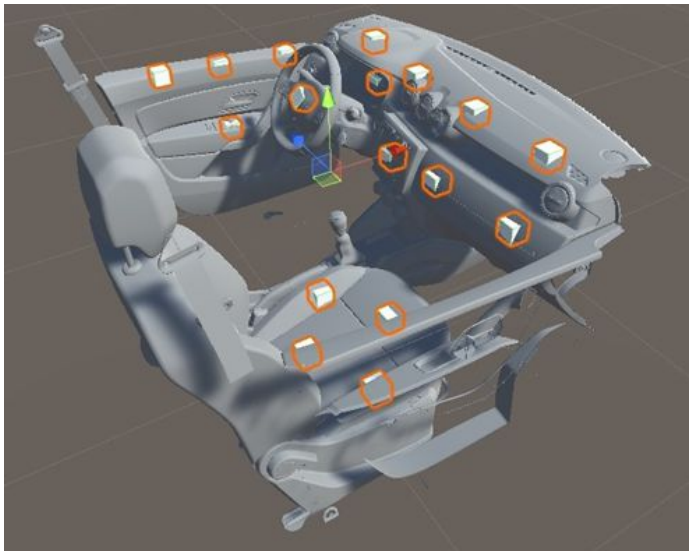

FIGURE 5. Location of points of interest in the interior of the car

Formally a k-d tree is a balanced binary tree for a set of data points $p_1, p_2, ..., p_n \in P$. where the root corresponds to all points and its two children represent almost equal-sized subsets of P. Every leaf corresponds to a k-dimensional point and every non-leaf node is a splitting point generating a hyper plane that splits points into subsets in a level-wise manner. Points to the left of this hyper plane are represented by the left sub-tree of the node and points to the right are represented by the right sub-tree. For a given node p at level i, the points associated with p are split into two halves by resorting to the median in dimension i mod k. Such that the point inserted in the tree at each step is the one, which has the median coordinate in the direction considered [3]. However, splitting rules may vary. The recursive construction ends as soon as a node p corresponds to a singleton or to a set of predefined sizes.

3.4.2 Nearest Neighbor Search using the k-d tree

Given a point q, find the point p in the data set P that is closest to q. This can be done by the 3 following steps :
Step 1: We defined the cubes representing the position and orientation of the surface for each of the ROI in the car model.
Step 2: Build a k-d tree to store the positions for all the cubes.
Step 3: Using the hand position as a query, we find the closest point to the hand from the k-d tree.
The algorithm 1 describes precisely the step 3.

Algorithm 1. Nearest Neighbor Search

Input: k-d tree root node, query point q

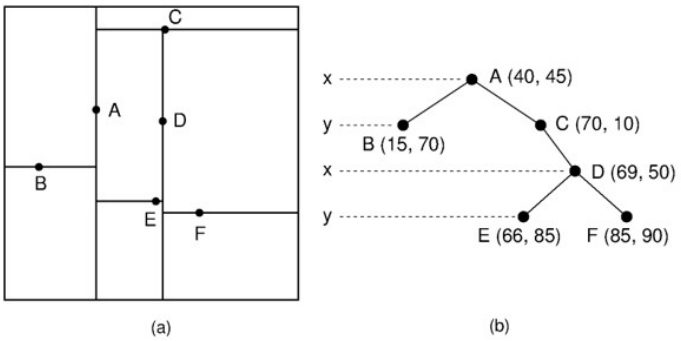

FIGURE 6. k-d tree (a) k-d tree decomposition for point set and (b) The resulting k-d tree.

Output: Nearest node p

1. Start from the root node.
2. Move down the tree recursively following the same procedure as it would be for the insertion of point q in the tree.
3. Once a leaf node is reached, save the leaf as the current best point p.
4. Rewind the recursion tree, and
5. For each node v, If $d(q,p) > d(q,v)$, then current node $v = p$, where d is a distance metric.
6. Check if there could be yet better points on the other side of the subtree by checking if neighboring boxes potentially contain points that are closer to q as the current best candidate (using the median values).
7. In case a point might be closer, recurse to the sub-tree that has not yet been visited.
8. If there could be, move down again on the other side of the sub-tree. Otherwise, go up another level.

3.4.3 Drawbacks of the Above Approach

The above approach has the following shortfalls:

1. The accuracy decreases when the hand is close to the midpoint of any two points. In this case, the difference in the distance between each point and the hand is very small and such that a slight displacement of the hand results to inappropriate motion of the robot to any of the nearest points.
2. Unnecessary and Involuntary hand movements. Due to human nature, the user can move their hands involuntarily without intention to interact with any objects in the space. In this case, the algorithm would still move the robot to the best point according to the minimum distance.

To overcome the above shortfalls, we decided to include the head gaze in the model such that a predicted point is considered valid and intentional. If the user was gazing in the direction of the object the hand interacts with. This is because when humans reach to grasp an object, they look at the target first, then bring the hand to the center of gaze as the object [19, 25, 21, 26].

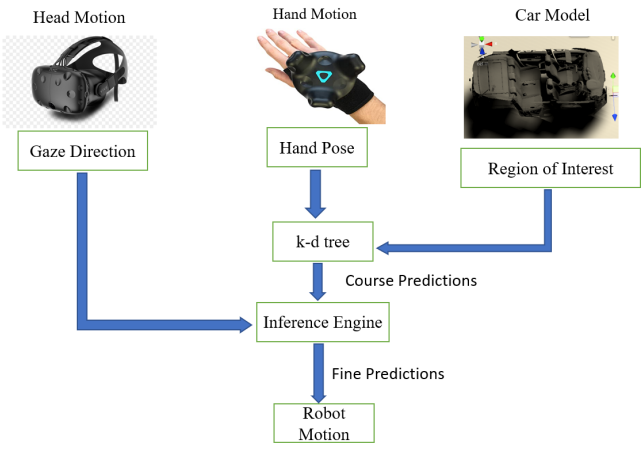

FIGURE 7. Proposed schematic diagram

3.4.4 Proposed Model

The proposed intention inference model from the regions of interest is summarized in Figure 7 as a 4 step process explained in algorithm 2.

Step 1: We defined the cubes representing the position and orientation of the surface for each of the ROI in the car model.

Step 2: Build a k-d tree to store the positions for all the cubes.

Step 3: Using the hand position as a query, we find the n_p points closest to the hand from the k-d tree. Contrary to section 3.4.2, we do not define directly the closest point but we select n_p candidate points. Depending on experiments n_p can be two or four.

Step 4 : From the n_p candidate points, we select only the n_{pg} points. belonging to the view frustum of the HMD.

Step 5: Among these n_{pg} points, the closest to the gaze direction is selected as predicted contact point between the user and the robot end effector. This selection is based on a distance from a point to a line. A detailed explanation is given in Section Appendix.

Theses steps are commented in the Algorithm 2.

Algorithm 2: Predictions with head gaze

Input: Scene information, head gaze direction
Output: position and orientation of desired point.
Method:

1. Cubes representing the surface and orientation of regions of interest.
2. Build a k-d tree for all points in the scene.
3. Using the hand pose as a query point q, return the nearest

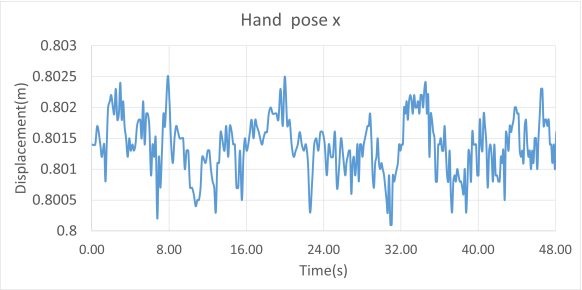

FIGURE 8. Hand Tracker Motion

Robotpose1.png

FIGURE 9. Robot TCP motion without head gaze

n_p-points from the k-d tree.

4. For each of the n_p points, we select only the n_{pg} points which lie within the view frustum of the HMD.
5. find the gaze direction as a unit vector from the central point of the eyes and draw a ray in the gaze direction.
6. calculate the distance of each point in n_{pg} from the ray and return the position of the nearest point. The equation of calculating the distance between a point and a line is equation (8) defined in the appendix section.

3.4.5 Experimentation

We conducted experiments to analyze the motion of the robot in response to hand motion in two approaches by moving the hand between the two points, with the following objectives:

1. Midpoint test: To show the response and error in robot motion when the hand is close to the midpoint. For this, we consider two points, which are 1.3 meters apart. Placing the hand tracker at the midpoint and displacing it by 2 cm results in the robot going to extreme points
2. Involuntary and Unintended Hand Motion Test: To verify that the robot moves only when the hand and head gaze are in the same direction. A hand motion to areas outside human vision is not sufficient to move the robot.

3.4.6 Results

In the first approach, we used the model without data from VR HMD, we observed that by moving the hand approximately 1.5 mm from the midpoint, the robot responds by moving to the other closer point. Figure 8 shows the displacement of the hand. The corresponding motion of the robot is shown in Figure 9. It can be seen that Robot TCP moves to the extreme points evertime there is a small displacement in the hand tracker. To avoid this noise motion we introduce Head Gaze.

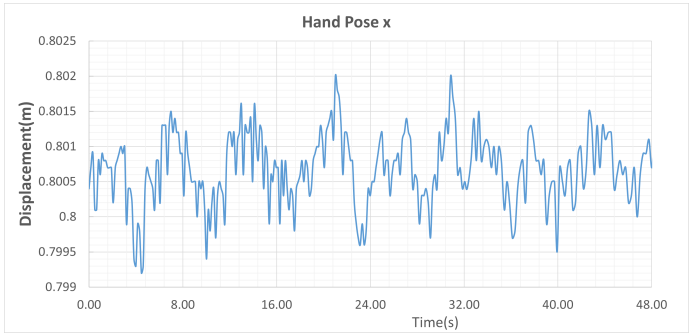

FIGURE 10. Hand Tracker Motion

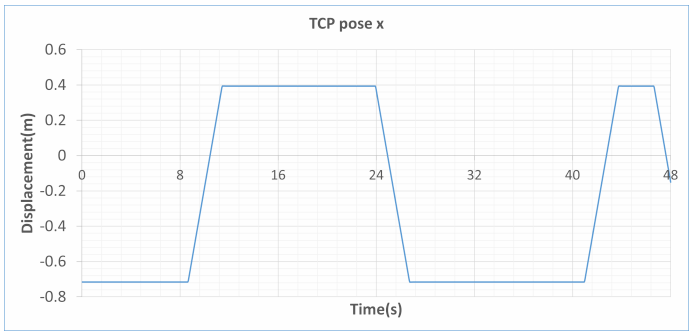

FIGURE 11. Robot TCP motion with head gaze

In the second approach, by including head gaze in the model, it is observed that the robot only moves in the direction of the object where head is currently facing. While Figure 10 shows displacement of the hand tracker about the mid point, it can be noticed that there is no significant change in the displacement of the tracker. However, for the robot TCP, it changes only when the gaze shifts. Figure 11 shows motion of the robot TCP, and when comparing it with Figure 9, we can see a reduced noise on displacement of robot.

From Figures 11 and 10, it was observed the movement of the hand does not affect motion of the robot as long as the head is not facing in the direction of the hand movement. The robot only changes direction of motion at instances corresponding to the rotation of the head.

The fact that gaze direction is taken into account limits the inappropriate variations of the target point and allows a smoother movement of the robot. This is based on the assumption that the operator looks in the direction where he wants to go [6-8].

4 Different Velocity Zones

The robot must be moved closer to the target point to prepare for the interaction. The movement must be fast so that the robot has arrived before the human and thus avoid unpleasant waiting

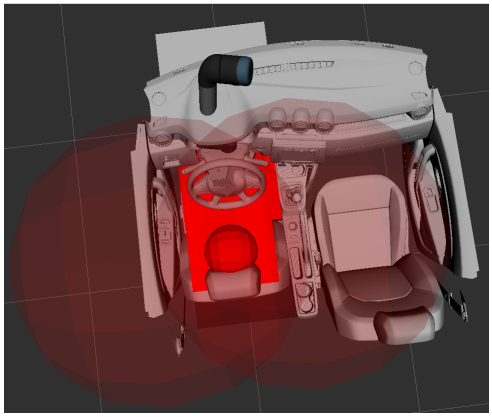

FIGURE 12. Interior of Car and User Workspace

but the maximum speed of the robot must be limited for safety reasons. Fig. 12 shows the scene of the VR environment, it consists of car interior and user model. Based on this, we distinguish three zones:

- WH: The human workspace, defined as 2 spheres whose radius is the size of the arm centered on the shoulders of the mannequin. This workspace will evolve according to the movements of the human. We could also consider a constant space if we limit the realistic movements of the torso. This space is represents by blue circles in Fig 13.
- IC: The inside of the car: this space delimits the area where we know the human must move. Even if the unity model is complex, this zone can be approximate by a larger simple region that includes the real interior of the car. The gray rectangle, in general, represents the entire Unity Model and we define a plane, depicted by the red line in Fig 13, that separates the region that can be reached by the user.
- EL: free space, which corresponds to what is neither in WH nor in IC. We can have a certain safety margin to define this zone. In our example, this zone is simply limited by a plane represented in red in Fig 13.

The limit on the robot velocity is chosen according to space:

- When the robot moves in EL, it can do so at maximum speed V_m (all parts of the robot are in EL) ,
- When the robot moves outside of EL, it must move at reduced speed V_r,

The speeds are chosen such as $V_m \geq V_r \geq 0$. The different spaces are shown in Figure 13. The blue hollow circle is the robot workspace, two blue-filled circles are the workspace of the user's hands. The grey rectangle is the complete interior model of the car and the red line is the plane that we use to differentiate the reachable and unreachable parts of the car by the user.

The robot is moving in the Cartesian space in a straight line to manage more easily the changes of space. We use a temporary

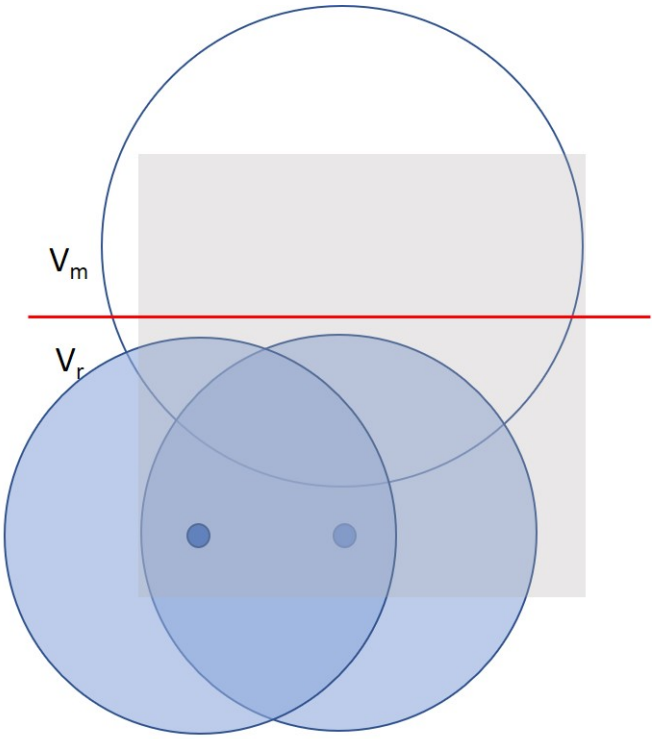

FIGURE 13. The different spaces defining the robot velocity

evolution defined by a trapezoidal shape to have the best compromise between travel time and maximum speed.

We do not discuss the choice of the correct max accelerations and the maximum speeds in Cartesian space, because we assume that the safety constraints are the strongest.

5 Velocity Profiles Based on Zones

The speed of cobot motion defined in the ISO standard for human-robot collaboration is 0.3 m/s. But such a low speed affects the response time for the robot to position itself to interact with the user.

So to consider safety and also have better response time we defined two velocity profiles of 0.3 m/s and 0.6 m/s. These are the V_m and V_r based on the zones defined above.

For this analysis, we devise the scene as shown in Fig 14. We define six points:

- 2 green points A and D are outside the plane boundary, these are 2 start/ home positions for the robot. Multiple home/start points are defined to have better access to points closer to each home pose and reduce the response time.
- 2 grey points B and C are inside the plane boundary, one point on the steering wheel and another on the tablet screen.

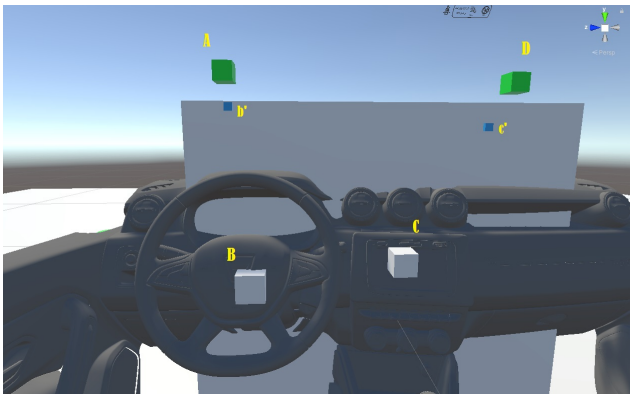

FIGURE 14. The Definition of Points in Interior of Car in Unity

- 2 blue points b' and c' on the plane boundary, these points are defined by calculating the intersection of the line with the plane for the robot's Cartesian trajectory from A to B and D to C respectively.

We analyze three different scenarios based on different velocity combinations.

- Scenario 1: A constant velocity of 0.25 m/s is used to define the motion throughout the zones. No velocity profiles are used based on zones. Motion is performed between points A, B, C, and D. In this scene the intersection points b' and c' are not considered.
- Scenario 2: Two velocities 0.25 m/s and 0.6 m/s are chosen based on the zones. If the robot travels from outside to the inside of the plane or vice-versa, we choose the low velocity (0.25 m/s). If the robot is moving outside the plane, we choose high velocity (0.6 m/s). Motion is performed between points A, B, C, and D. In this scene also the intersection points b' and c' is not considered.
- Scenario 3: In this scene, we have the same velocity profile as in scenario 2, but in this scene, the intersection points b' and c' are considered. This changes the motion points for moving from A to B, to A to b', and from b' to B. It is the same for the trajectories that have to cross the plane. Moving from C to D, using intermediate point c'.

Each scenario has the same five motions:

- From A to B, moving from outside to inside the plane
- From B to C, moving inside the plane
- From C to D, moving from inside to outside the plane
- From D to A, moving outside the plane.
- From A to C, moving from outside to inside.

Table 1 presents velocities of motions in all three scenarios. Figure 15 shows the motion plot of TCP for all the selected motions. A comparison between scenario 1 and scenario 2 can give us that robot moves faster when it is outside the plane. However,

Motion	Scenario 1	Scenario 2	Scenario 3
From A to B	0.25	0.25	A to b' 0.25 b' to B 0.6
From B to C	0.25	0.25	0.25
From C to D	0.25	0.25	C to c' 0.25 c' to D 0.6
From D to A	0.25	0.6	0.6
From A to C	0.25	0.25	A to b' 0.6 b' to C 0.25

TABLE 1. Different velocities based on motion in the three scenarios

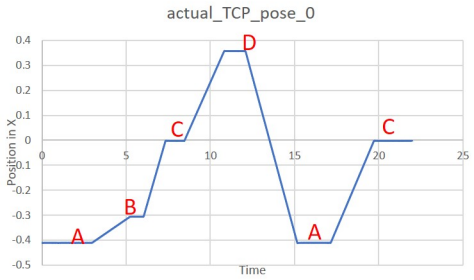

FIGURE 15. All Motions Together

there is no significant change in velocity when the robot is moving towards or inside the plane. This is where scenario 3 comes to light. When the robot moves from an outside point to an inner point, scenarios 1 and 2 have the same speeds, but in scenario 3, due to the intersection points, a high speed is used partially in the movement.

A comparison of all three scenarios for Motion from A to B is shown in Figure 16. It can be seen that for scenario 1 and scenario 2 there is no significant change in time. But in scenario 3 due to high velocity till the intersection point, we can see, the robot reaches its position 0.1 s earlier than in previous scenarios. These tests done are experimental, thus in some cases, the desired velocity was not reached because the duration of the phase is too short.

6 Trajectory Planning of Robot

So far, for the velocity analysis, we have performed the trajectory in the Cartesian plane. This implies the robot moves in a straight line. To define the path from the current robot posture to the target posture, two strategies are compared:

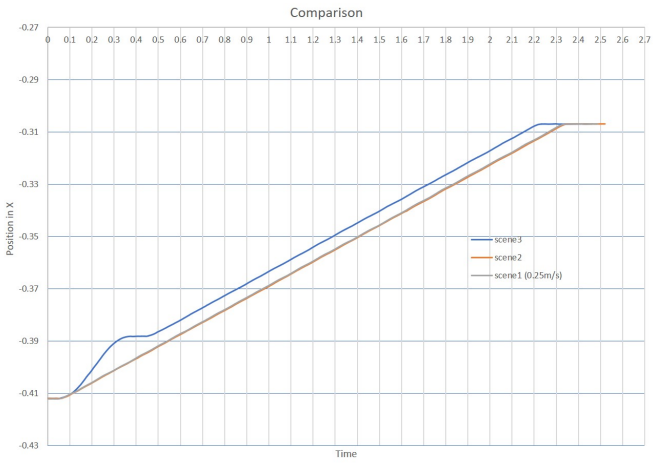

FIGURE 16. A Comparison of Motion 1 for all 3 scenarios

FIGURE 17. The Test points used for Trajectory Planning

- *the shortest way*: Knowing the target point, the robot moves towards it, and adapts its speed according to the spaces it crosses,
- *via points*: use of intermediate points to keep the robot's speed high. We define intermediate points to minimize the distance traveled in the area with the lowest speed, or we can define intermediate points on a global time minimization criterion (approximate criterion assuming constant speed per section for example).

Figure 17 shows the interior of the car. Three points A, B and C are defined in the steering wheel, tablet screen, and glove compartment respectively. A plane can also be seen as discussed in the previous section to separate the user workspace in the interior of the car. There are also three points a', b', and c' which are the projection of desired points onto the plane.

In the previous scenarios, we had performed motion inside the car with reduced velocities. Nevertheless, in some cases when the desired point is in far exteriors of user space and it

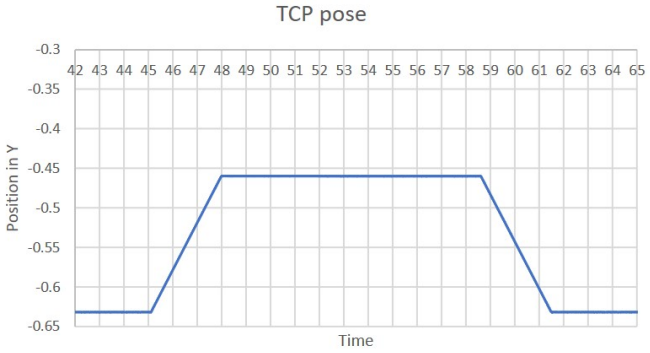

FIGURE 18. The path is a straight line between the initial and target pose

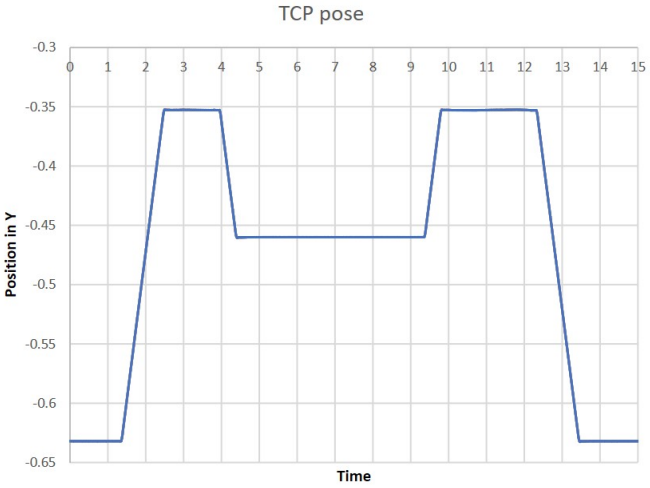

FIGURE 19. Using via points to maximize the path at fast velocity.

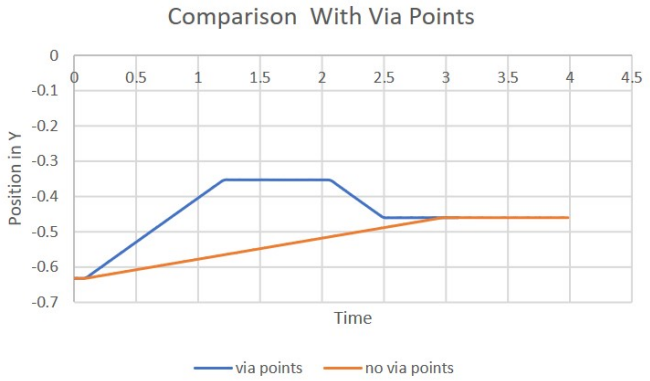

FIGURE 20. Comparison of using via points

takes more time to reach it due to low velocity. For such situations, we use the projection points to move the robot in high velocity to reach the target.

Therefore, if the robot has to move from A (steering wheel) to C (glove compartment), there are two possible trajectories.

- *Option 1*: To go from A to C with reduced velocity. As both the points are inside the plane and user workspace.
- *Option 2*: Start from A to a' with reduced velocity. Go from a' to c' with high velocity since this point is on the plane and the robot is moving outside the plane. Finally from c' to C move in reduced velocity to reach the destination.

In the plot shown in Figure 18, the robot moves from point A to C and going back to A, inside the plane with low velocity. It takes 3.1 s to move from A to C. This results in a reduced velocity profile and a long time to travel between the points.

Figure 19 shows the TCP pose for the same points A and C but using the Via-points a' and c'. Motion is from A to a', a' to c' and c' to C. Stop for 4 s and start from C to A via c' and a'. By using higher velocity to move between projection points the travel time is reduced.

A comparison of a trajectory between two points A to C located inside the car, with or without via points shows that when using via points a' and c', the duration of the trajectory is shorter. Figure 20 shows the time analysis for reaching the same points but with a different trajectory.

It is clear that in the case when not using via-points the time taken is 3.1 s. When considering the via-points it takes about 2.5 s to reach the destination.

7 Conclusions

In this article, a collaborative robot is used as a haptic interface with intermittent contact. A motion prediction algorithm is used to select the areas with which the user intends to interact and to move the robot as fast as possible while ensuring user safety.

We introduce two speed profiles for the user's safety. The robot moves at a higher speed when it is outside the user's workspace. In situations where there is a large distance between two points within the workspace, we introduce via points to reduce travel time. The time needed to add via points is less than the time needed to go directly inside the car while being much safer.

In future work, the aim is to add more zones to have multiple velocity profiles. To limit the reduced velocity V_r to the user workspace rather than inside the plane. So to introduce a new velocity V_i, such that $V_m \geq V_i \geq V_r \geq 0$. This new velocity would be in the region, which is inside the plane not covered by the workspace of the user.

Acknowledgement

This work was funded under the LobbyBot project, ANR-17-CE33 [1]. The authors of the article thank the members of the project for their help in carrying out this work, Lionel Dominjon, Sandrine Wullens, Allexandre Bouchet, Javier Posselt, Anatol Lecuyer and Victor Rodrigo Mercado Garcia.

Appendix

The distance from a point to a line is the shortest distance from a given point to any point on an infinite straight line. It is the perpendicular distance of the point to the line.

Formally, a line in three dimensions is specified by two points $p_1 = (x_1, y_1, z_1)$ and $p_2 = (x_2, y_2, z_2)$ through which it passes. A vector v along the line is given by

$$v = \begin{bmatrix} x_1 + (x_2 - x_1)t \\ y_1 + (y_2 - y_1)t \\ z_1 + (z_2 - z_1)t \end{bmatrix} \tag{1}$$

The squared distance d between a point on the line and our point of interest $p_i = (x_i, y_i, z_i)$ is given by

$$d^2 = [(x_1 - x_i) + (x_2 - x_1)t]^2 + [(y_1 - y_i) + (y_2 - y_1)t]^2 + [(z_1 - z_i) + (z_2 - z_1)t]^2 \tag{2}$$

Our objective is to minimize d, we set $d(d^2)/dt = 0$ and solve for t to get

$$t = -\frac{(p_1 - p_i) \cdot (p_2 - p_1)}{|p_2 - p_1|^2} \tag{3}$$

where $\cdot$ denotes the dot product.
Substituting t into equation (2), we obtain

$$\begin{aligned} d^2 = {} & [(x_1 - x_i)^2 + (y_1 - y_i)^2 + (z_1 - z_i)^2 \\ & + 2t[(x_2 - x_1)(x_1 - x_i) + (y_2 - y_1)(y_1 - y_i) + (z_2 - z_1)(z_1 - z_i)] \\ & + t^2[(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2] \end{aligned} \tag{4}$$

which simplifies to

$$d^2 = |p_1 - p_i|^2 - 2\frac{[(p_1 - p_i) \cdot (p_2 - p_1)]^2}{|(p_2 - P_1)|^2} + \frac{[(p_1 - p_i) \cdot (p_2 - p_1)]^2}{|(p_2 - p_1)|^2} \tag{5}$$

$$d^2 = \frac{|p_1 - p_i|^2 |p_2 - p_1|^2 - [(p_1 - p_i) \cdot (p_2 - p_1)]^2}{|(p_2 - p_1)|^2} \quad (6)$$

$$d^2 = \frac{|(p_2 - p_1) \times (p_1 - p_i)|^2}{|p_2 - p_1|^2} \quad (7)$$

where $\times$ denotes the cross product.
The value of d is obtained by taking the square root in (7)

$$d = \frac{|(p_2 - p_1) \times (p_1 - p_i)|}{|p_2 - p_1|} \quad (8)$$

REFERENCES

[1] Lobbybot project. https://www.lobbybot.fr/. Accessed: 2021-05-21.

[2] Bruno Araujo, Ricardo Jota, Varun Perumal, Jia Xian Yao, Karan Singh, and Daniel Wigdor. Snake charmer: Physically enabling virtual objects. In *Proceedings of the TEI'16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction*, pages 218–226. ACM, 2016.

[3] Jon Louis Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM*, 18(9):509–517, sep 1975.

[4] Andrea Cherubini, Robin Passama, André Crosnier, Antoine Lasnier, and Philippe Fraisse. Collaborative manufacturing with physical human–robot interaction. *Robotics and Computer-Integrated Manufacturing*, 40:1–13, 2016.

[5] Takahiro Endo, Haruhisa Kawasaki, Tetsuya Mouri, Yasuhiko Ishigure, Hisayuki Shimomura, Masato Matsumura, and Kazumi Koketsu. Five-fingered haptic interface robot: Hiro iii. *IEEE Transactions on Haptics*, 4(1):14–27, 2011.

[6] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional Two-Stream Network Fusion for Video Action Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, jun 2016.

[7] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Trans. Math. Softw.*, 3(3):209–226, sep 1977.

[8] Philippe Fuchs, Guillaume Moreau, and Pascal Guitton. *Virtual reality: concepts and technologies*. CRC Press, 2011.

[9] V.K. Guda, D. Chablat, and Chevallereau C. Safety in a human robot interactive: Application to haptic perception. In *International Conference on Human-Computer Interaction HCII 2000*, 2020.

[10] Mary Hayhoe, Pilar Aivar, Anurag Shrivastavah, and Ryan Mruczek. Visual short-term memory and motor planning. *Progress in brain research*, 140:349–363, 2002.

[11] Roland S. Johansson, Göran Westling, Anders Bäckström, and J. Randall Flanagan. Eye–hand coordination in object manipulation. *Journal of Neuroscience*, 21(17):6917–6932, 2001.

[12] Haruhisa Kawasaki and Tetsuya Mouri. Design and control of five-fingered haptic interface opposite to human hand. *IEEE Transactions on Robotics*, 23(5):909–918, 2007.

[13] Yaesol Kim and Young Kim. Versatile encountered-type haptic display for vr environment using a 7-dof manipulator. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'2016)*, 2016.

[14] Hema S. Koppula, Ashesh Jain, and Ashutosh Saxena. Anticipatory planning for human-robot teams. In *Springer Tracts in Advanced Robotics*, volume 109, pages 453–470. Springer Verlag, 2016.

[15] Oscar La De Cruz, Florian Gosselin, Wael Bachta, and Guillaume Morel. Contributions to the design of a 6 dof contactless sensor intended for intermittent contact haptic interfaces. In *2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 130–135. IEEE, 2018.

[16] Philip Long, Christine Chevallereau, Damien Chablat, and Alexis Girin. An industrial security system for human-robot coexistence. *Industrial Robot: An International Journal*, 45(2):220–226, 2018.

[17] Ruikun Luo, Rafi Hayne, and Dmitry Berenson. Unsupervised Early Prediction of Human Reaching for Human—Robot Collaboration in Shared Workspaces. *Auton. Robots*, 42(3):631–648, mar 2018.

[18] William A. McNeely. Robotic graphics: a new approach to force feedback for virtual reality. In *Proceedings of IEEE Virtual Reality Annual International Symposium*, pages 336–341. IEEE, 1993.

[19] Víctor Rodrigo Mercado, Maud Marchal, and Anatole Lécuyer. Entropia: Towards infinite surface haptic displays in virtual reality using encountered-type rotating props. *IEEE Transactions on Visualization and Computer Graphics*, 2019.

[20] Land Michael, Mennie Neil, and Rusted Jennifer. The roles of vision and eye movements in the control of activities of daily living. *Perception*, 28(11):1311–1328, 1999. PMID: 10755142.

[21] Jeff Pelz, Mary Hayhoe, and Russ Loeber. The coordination of eye, head, and hand movements in a natural task. *Experimental Brain Research*, 139(3):266–277, 2001.

[22] C. Pérez-D'Arpino and J. A. Shah. Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification. In *2015 IEEE International Conference on Robotics and Automa-*

tion (ICRA), pages 6175–6182, may 2015.

[23] Jérôme Perret and Pierre Vercruysse. Advantages of mechanical backdrivability for medical applications of force control. In *Conference on Computer/Robot Assisted Surgery (CRAS)*, pages 84–86, 2014.

[24] H. C. Ravichandar and A. Dani. Human intention inference and motion modeling using approximate E-M with online learning. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1819–1824, 2015.

[25] Steeven Villa Salazar, Claudio Pacchierotti, Xavier de Tinguy, Anderson Maciel, and Maud Marchal. Altering the stiffness, friction, and shape perception of tangible objects in virtual reality using wearable haptics. *IEEE Transactions on Haptics (ToH)*, 2020.

[26] Jeroen B. J. Smeets, Mary M. Hayhoe, and Dana H. Ballard. Goal-directed arm movements change eye-head coordination. *Experimental Brain Research*, 109(3):434–440, 1996.

[27] Alexander Stamenkovic. Do postural constraints affect eye, head, and arm coordination? *Journal of neurophysiology*, 120(4):2066–2082, 2018.

[28] Marija Tomić, Christine Chevallereau, Kosta Jovanović, Veljko Potkonjak, and Aleksandar Rodić. Human to humanoid motion conversion for dual-arm manipulation tasks. *Robotica*, 36(8):1167–1187, 2018.