2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Physical Human-Robot Interaction with Real Active Surfaces using Haptic Rendering on Point Clouds

Michael Sommerhalder[1,2], Yves Zimmermann[1,2], Burak Cizmeci[1], Robert Riener[2,†], and Marco Hutter[1,†]

***Abstract*— During robot-assisted therapy of hemiplegic patients, interaction with the patient must be intrinsically safe. Straight-forward collision avoidance solutions can provide this safety requirement with conservative margins. These margins heavily reduce the robot's workspace and make interaction with the patient's unguided body parts impossible. However, interaction with the own body is highly beneficial from a therapeutic point of view. We tackle this problem by combining haptic rendering techniques with classical computer vision methods. Our proposed solution consists of a pipeline that builds collision objects from point clouds in real-time and a controller that renders haptic interaction. The raw sensor data is processed to overcome noise and occlusion problems. Our proposed approach is validated on the 6 DoF exoskeleton ANYexo for direct impacts, sliding scenarios, and dynamic collision surfaces. The results show that this method has the potential to successfully prevent collisions and allow haptic interaction for highly dynamic environments. We believe that this work significantly adds to the usability of current exoskeletons by enabling virtual haptic interaction with the patient's body parts in human-robot therapy.**

I. Introduction

Rehabilitation robots were strongly researched on during the last years, and they start getting employed commonly for therapy. In these therapies, physical human-robot interaction is a key challenge to guarantee reliable, safe, and successful therapies. Since such trainings include an exoskeleton robot, a patient, and a therapist, it creates a highly dynamic environment that needs to be perceived in real-time and be incorporated into the robot's control framework to guarantee a safe therapy.

Activities including interaction with the head, face, and second arm are essential in daily life and are often trained in conventional therapy. However, since most robotic solutions use conservative static collision boundaries that decrease the robot's workspace, as well as restricting designs of their devices, these activities cannot be trained with the robot. This open problem reduces the usability of robots in rehabilitation therapies and needs to be addressed.

Through live observation of the relevant surfaces and a haptic rendering strategy, we want to enable body interaction while maintaining safety. Our proposed solution consists of a novel perception pipeline to find the closest and most likely collision point to the robot's end-effector. A trajectory controller then predicts the dynamics of the collision surface and up-samples the data, to meet the criterion for haptic force rendering. A positional and velocity constraint apply the collision surface in a hierarchical task controller to close the loop between perception and control. A generated point cloud of an example scene using an external camera system is shown in Fig. 1. We deploy this pipeline on ANYexo, an exoskeleton that is suited for proximal movements.

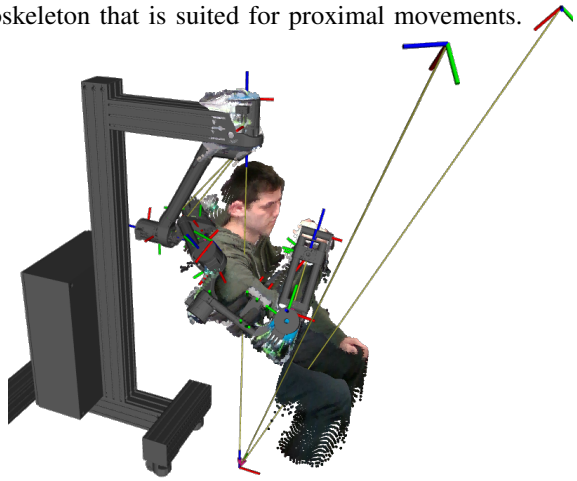

Fig. 1. ANYexo model (dark gray) with external RGB-D camera poses and user as recorded point cloud.

A. Related work

Collision avoidance and haptic interaction are well-studied fields in robotics. Many collision avoidance strategies are based on the idea of potential fields [1] [2] [3] [4]. These methods approximate the work space with a 3D grid and calculate the resulting force for each cell based on targets (attractive forces) and obstacles (repulsive forces). This information is usually processed by a trajectory controller that optimally avoids obstacles. Such methods are therefore well-suited for collision avoidance in general. But since they usually maximize the distance to any obstacle, these methods are unsuited for our stated problem. Obstacle avoidance approaches using RGB-D point cloud data, as coming from Kinect or Realsense depth sensors, are also well-studied. The authors in [5] extend successful systems for such data. Latter examines collision avoidance by calculating reaction forces on a 7-DOF Kuka robot using a voxel map approach. Knopp et al. [6] then use such robot as a haptic device. They realize that the KUKA LBR iiwa robot is only viable for haptic scenarios which require high forces and

This research was supported in part by the Swiss National Science Foundation through the National Centre of Competence in Research Robotics (NCCR Robotics) and Innosuisse, the Swiss Innovation Agency.

[1] M. Sommerhalder, Y. Zimmermann, B. Cizmeci and M. Hutter are with Robotic Systems Lab, ETH Zurich, Switzerland yvesz@ethz.ch

[2] M. Sommerhalder, Y. Zimmermann and R. Riener are with Sensory-Motor Systems Lab, ETH Zurich, Switzerland. R. Riener is additionally with Spinal Cord Injury Center, University Hospital Balgrist, Zurich, Switzerland riener@ethz.ch.

[†] R. Riener and M. Hutter contributed equally to the project's lead.

torques, since the robot's inertia lead to high drag and, in interaction with the user, to slight oscillations of the end-effector. ANYexo [7], a versatile exoskeleton based on series elastic actuation, overcomes this problem by its light-weight hardware design. Advancements in haptic force rendering, such as calculating reaction forces with a spring-damping system upon penetration of the object [8], or using neural networks [9] to determine repulsive forces, build on the work by [10] and [11]. Salisbury et al. [10] determines the required update frequency for smooth haptic interaction and proposes a ground lying architecture, while [11] presents a method on how to successfully render obstacles with adequate sense of touch. [8] does not take occlusion problems into account, and [9] only considers static collision objects. The authors in [12] extend the initial methods to streaming (unfiltered) point cloud data and improve the robustness of the slip-through problem for the proxy object. Our method, as described in section IV-E, extends this method to retrieve collision surface information, and increases the robustness for disturbed point clouds by approximating the surface with multiple concentric circles. The authors in [13] use these techniques to haptically render forces on RGB-D data for a surgical robot on a linked Omni haptic device.

For the incorporation of boundary constraints in the hierarchical task controller, Hutter et al. [14] show the ground lying controller concept that uses null-space projection for stacking prioritized operational space tasks. Bellicoso et al. [15] builds upon this idea and shows how various prioritized constraints such as contact forces can be built and added for legged robots, while [16] show how such constraints can be used in haptic rendering problems, and [17] uses a constraint-based collision avoidance approach by defining appropriate inequality constraints for obstacles.

Based on above-mentioned related work, our main contribution consist of (1) a novel processing step to overcome the occlusion problem, especially occlusions of unguided body parts by the robot, (2) a novel method that extends [12] to increase robustness for partially occluded and noisy surfaces using concentric circle approximation, (3) a novel sequence and configuration of five consecutive computer vision processes to create robust collision surfaces from noisy RGB-D data, and (4) positional and velocity constraints for haptic rendering in hierarchical task control.

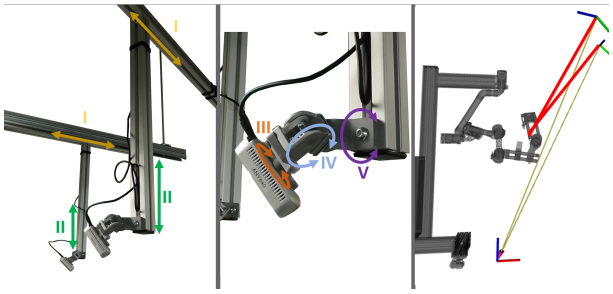

Fig. 2. Mounting of two RGB-D sensors to detect and avoid collisions between ANYexo and the patient, while allowing safe interaction. The setup has 5 degrees of freedom (I,II are prismatic, and III-V are rotational joints), to fine-tune the camera's optimal pose.

II. System Architecture

Two cameras record the patient and exoskeleton, as shown in Fig. 2. The collision avoidance and haptic rendering pipeline is split into the following sub-systems: (1) Perception, (2) Data Processing, (3) Haptic Force Rendering and (4) Hierarchical Task Controller. Fig. 3 shows the pipeline with a detailed view at the data processing step, whose components are explained in detail in section III. The raw point cloud of each sensor at time step k, $P^k_{\text{raw},1}$ and $P^k_{\text{raw},2}$, is fed into our data processing pipeline. Resulting closest collision point and surface normal vector c^k, n^k are translated into a positional α^k_P, velocity α^k_V and force constraint α^k_F. These constraints are added to a hierarchical task controller that calculates the final joint torques τ_J for the exoskeleton.

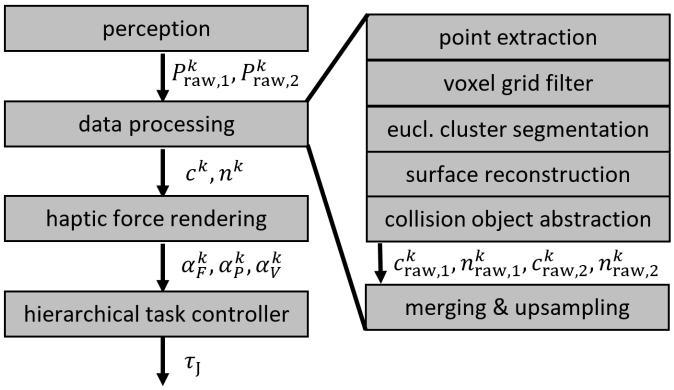

Fig. 3. System architecture and detailed data processing pipeline. The pipeline is applied to each sensor input, before the estimations are merged and up-sampled to be made ready for haptic interaction.

III. Perception

To define the extrinsic placement of the sensor system, it was important to consider the body parts that could potentially collide with the robot's end-effector. DIN ISO/TS 15066 [18] provides a list of maximal allowed impact forces on different body parts, for which a collision might lead to negligible injuries at most. Based on this information and the RoM of ANYexo, relevant body parts, for which collisions can occur and need to be avoided, were extracted and the cameras were mounted such that these areas, namely forehead, torso and left arm, are centered in the field of view of both sensors and occlusions are minimized.

Solutions involving proximity sensors that could be directly mounted on the robot's end-effector were thought of, but discarded due to cost and complexity reasons. As global information about the collision shape (head, torso) would be lost with proximity sensors, as only the local surface could be estimated, multiple arrays of these sensors would need to be necessary to achieve the same results as with two external RGB-D cameras.

Two cameras were finally mounted for the right-handed exoskeleton. The resulting camera frames can be seen in Fig. 1, while Fig. 2 shows the mounting hardware that enables 5 DoF fine-tuning of the camera's pose. We chose a configuration where one camera is placed in front of the

patient and one camera is placed at a 45° angle on the right side of the patient. This setup has the advantage of increasing robustness to our collision point estimation by having an overlapping field-of-view, and reducing occluded areas with its two different camera perspectives.

IV. Point Cloud Data Processing

The data processing pipeline is introduced in Fig. 3 and the processing steps can be seen in Fig. 4. We refer to our video [1] that shows the pipeline step by step. For each sensor, the point cloud undergoes identical processing steps. Each camera therefore provides a raw collision point estimate. In a final step, all estimates are merged, and the final collision point estimation is temporally filtered and up-sampled, to meet the requirement of having smooth haptic interaction that runs at 800 Hz.

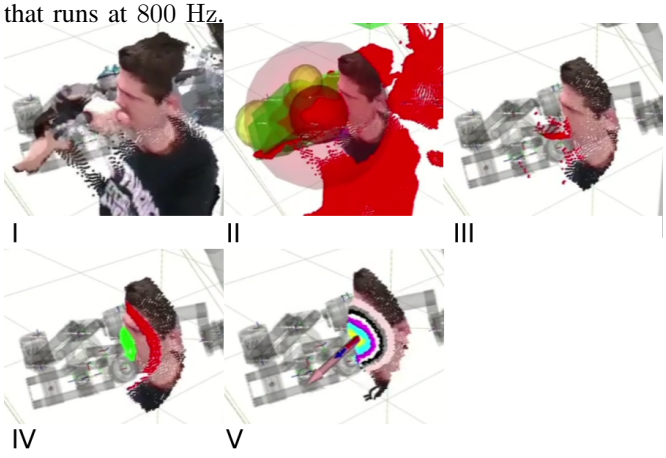

Fig. 4. Data processing pipeline, top left to bottom right: initial cloud, relevant point extraction, euclidean cluster segmentation, surface reconstruction, collision object abstraction, merging and up-sampling. Red parts in II and III denote rejected points. Red parts in IV are a selection of points for ICP matching, and green points indicate the reconstructed surface.

A. Extraction of Relevant Points

The aim of this first process is to remove most non-relevant data points from the initial cloud, such as background, to reduce the computational cost in the next steps. Points that belong to the exoskeleton and the impaired arm are removed as well, since points from the robot should not contribute to collision surface of an external obstacle. For computational efficiency, primitive objects are defined that can be attached to a robot's frame. For collision avoidance with the exoskeleton's end-effector as haptic interaction point (HIP), objects as illustrated in Fig. 4, upper-left image, are attached to the robot. Each cycle, all points that lie inside the primitives (or outside of the inverted semi-transparent red sphere) are removed.

B. Voxel Grid Filter

This small step down-scales the cropped point cloud to an average of 5000 points by approximating adjacent points with voxels. We use this step to increase the speed of the computationally expensive surface reconstruction and clustering steps later on.

[1] https://youtu.be/c488sOlY_6Q

C. Euclidean Cluster Segmentation

To remove points that do not belong to a distinct surface, we use PCL's Euclidean Cluster Segmentation [19], and neglect all clusters that have a point size lower than a certain threshold. With this method we can successfully prepare the data for the noise-sensitive surface reconstruction in the next step.

D. Surface Reconstruction

So far we handled data reduction and outlier removal problems with the first three processing steps. The next challenge is to recover the surface that is occluded by the robot as it approaches towards the human body. The occluded points are the most likely ones to contain rich information about the collision point. An example situation is shown in Fig. 5. For non-planar, convex surfaces such as the chin, this is especially important, since information about the surface's closest point to the robot is lost. Additionally, to consider sudden movements from the patient's unguided body parts, the surface reconstruction has to take the obstacle's movement into account.

This is solved by integrating a frame-to-frame ICP matching algorithm [20], as shown in Alg. 1. Fig. 6 visually explains the procedure that can be split into four main parts:

1) At step [k-1], all points of the reconstructed point cloud within r_{out} and r_{in} are stored in memory.
2) At step [k], the memorized points are ICP-matched on the points of the new cloud that lie within r_{in} and r_{out}
3) Points outside r_{in} are removed, and the remaining points merged to the new cloud.

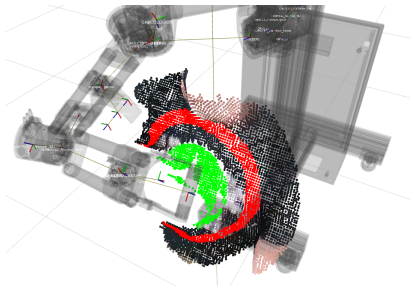

Fig. 5. The closest collision point is occluded by the robot end-effector. Our surface reconstruction algorithm fills the gap with memorized points (green) fitted to a selection of the current perception (red).

1) Limitations: Any outlier points as well as surface noise are not removed by this technique, but rather transported to the next frame due to the ICP step. This makes a careful outlier removal necessary, as described in section IV-C. Additionally, this step introduces a constraint on the relative velocity between the HIP and the haptic collision point for two consecutive frames:

For a time step Δt, the velocity component that is parallel to the normal direction of the approximated collision surface $v_{\perp}$ is bounded by

$$v_{\perp} \leq \frac{r_{\text{out}} - r_{\text{in}}}{\Delta t} \tag{1}$$

because intermediate points would not be seen by any of the two frames.

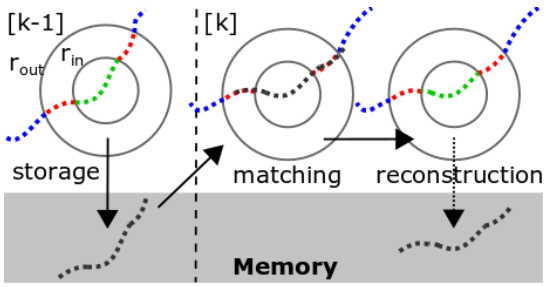

Fig. 6. Surface reconstruction algorithm: (I) extract points from frame k-1 (red) and add them to the memory (black), (II) ICP-match the memorized points to the cloud of time-step k; (III) remove points outside r_{in} and merge point set with current cloud.

Algorithm 1: SURFACE RECONSTRUCTION

Input: Point cloud P^k and HIP position h^k of time step k. Memory P^{k-1}_{extr} from last time step.
Output: Extracted and icp-matched point cloud P^k_{extr}

$P^k_1 \leftarrow \{p \in P^k \mid \text{dist}(p, h^k) < r_{\text{in}}\}$
$P^k_2 \leftarrow \{p \in P^k \mid \text{dist}(p, h^k) < r_{\text{out}}\}$
$P^k_{\text{extr}} \leftarrow P^k_2 \setminus P^k_1$
$P^k_{\text{icp}} \leftarrow \text{ICP}(P^{k-1}_{\text{extr}}, P^k_{\text{extr}})$
if $P^k_{\text{icp}} \neq \emptyset$ **then**
  $\tilde{P}^k_{\text{icp}} \leftarrow \{p \in P^k_{\text{icp}} \mid \text{dist}(p, h^k) < r_{\text{in}}\}$
  $P^k_{\text{extr}} \leftarrow \tilde{P}^k_{\text{icp}} \cup P^k_{\text{extr}}$
end if
$P^{k-1}_{\text{extr}} \leftarrow P^k_{\text{extr}}$

E. Collision Object Abstraction

To extract accurate information about the closest collision point and its surface normal, the idea is to build concentric spheres around the haptic interaction point (HIP) with radii $0 < r_{\text{in}} < \cdots < r_{i-1} < r_i < \cdots < r_{\text{out}}$. All points within R_i and R_{i-1} are approximated by a circle C_i, whose center point c^k_i and normal vector n^k_i can be estimated. For the innermost sphere with R_0, a plane n^k_0 is fitted to the points and the mean position is used as center point c^k_0. These centroid and normal vector estimations are then averaged together to form the final surface estimation. Fig. 7 shows a sketch of the basic idea.

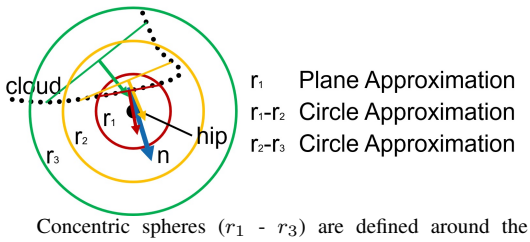

Fig. 7. Concentric spheres (r_1 - r_3) are defined around the haptic interaction point. All points lying between two consecutive spheres are approximated by a circle (green and yellow normal vector). The innermost points are approximated by a plane (red normal vector). The final surface is a weighted averaging of all estimates (blue normal vector n).

To further increase robustness, outlier estimates are filtered out by calculating the euclidean distance matrix and removing the worst half of the estimates from the ensemble. The algorithm to estimate c^k and n^k is described in Alg. 2.
Since the direction of the normal vector estimation is undefined, it can be determined by comparing it to the camera's normal direction z^k (see Eq. 2). This can only be done because we first estimate the collision point and only then we merge the estimates of multiple cameras.

$$\boldsymbol{n}^k_i \leftarrow \text{-sign}(\boldsymbol{n}^k_i \circ \boldsymbol{z}^k) \cdot \boldsymbol{n}^k_i \tag{2}$$

plane3d(P^k) and **circle3d**(P^k) are functions that calculate plane and circle estimations, respectively, of an input set of 3d-points. They are implemented by first estimating the normal direction using singular value decomposition, then transforming the problem to a planar space using Rodrigues formula, estimating the planar centroid using the QR decomposition for the linear least squares problem and transforming the solution back to 3D. The final estimates for c^k and n^k are estimated by averaging over the weighted circle estimates. They are weighted inverse proportional to their radius to weaken the influence of points with a big distance to the HIP.

Algorithm 2: OBJECT ABSTRACTION

Input: Point cloud P^k and HIP h^k of time step k, camera normal z^k
Output: Estimated centroid c^k and normal vector n^k.

for Sphere with Radius r_i, $r_i > r_{i-1}$ **do**
  $P^k_i \leftarrow \{p \in P^k \mid r_{i-1} < \text{dist}(p, h^k) < r_i\}$
  if $P^k_i \neq \emptyset$ **then**
    if P^k_i is first non-empty set **then**
      $[c^k_i, n^k_i] \leftarrow \text{plane3d}(P^k_i)$
    else
      $[c^k_i, n^k_i] \leftarrow \text{circle3d}(P^k_i)$
    end if
    $n^k_i \leftarrow \text{-sign}(n^k_i \circ z^k) \cdot n^k_i$
  end if
end for
$C^k \leftarrow \langle c^k_1 \dots c^k_n \rangle$
$N^k \leftarrow \langle n^k_1 \dots n^k_n \rangle$
$S^k \leftarrow \langle \sum_{i \neq 1} \text{dist}(n^k_1, n^k_i), \dots, \sum_{i \neq N} \text{dist}(n^k_N, n^k_i) \rangle$
$I^k \leftarrow \text{indicesFromQuickSort}(S^k)$
$I^k \leftarrow \langle i^k_1, \dots, i^k_{N/2} \rangle$
$n^k \leftarrow \text{weightedAvg}(N^k)$
$c^k \leftarrow \text{weightedAvg}(C^k)$

1) Limitations: The algorithm assumes to have one distinct closest collision point for the HIP h^k. If two equally distant points are visible, the circle and plane estimation fails and assumes a point between the two. One way to avoid this problem is to keep only the biggest visible point cluster in the Euclidean Cluster Segmentation step. Fortunately, when dealing with convex surfaces, as it is the case for the human body, there is usually exactly one distinct closest point.

F. Merging and Up-sampling

This step combines the centroid and normal estimates of each camera to make a final prediction about the collision surface. This is achieved in two steps:

1) Predict the future target position based on an estimated trajectory from the previous measurements
2) Update the current value of the centroid and normal vector using PID-control

As a regression model, we use a polynomial of 2nd degree to up-sample the data:

$$x(t) = a_2 t^2 + a_1 t + a_0 \tag{3}$$

The parameters a_2, a_1 and a_0 are calculated by solving the following linear system of equations:

$$(WA)x = (Wb), \quad x = \{a_0, a_1, a_2\}^T \tag{4}$$

with

$$A = \begin{pmatrix} 1 & t_1 & t_1^2 \\ \vdots & \vdots & \vdots \\ 1 & t_N & t_N^2 \end{pmatrix}, \quad b = \begin{pmatrix} c_i \\ \vdots \\ c_N \end{pmatrix} \tag{5}$$

and W being a diagonal weighting matrix that weights the measurements inverse proportional to their arrival time compared to the last measurement, and $\{c_1, \ldots, c_N\}$ being the observed measurements of coordinate c for the duration d. That means that the measurements b are coming from multiple cameras as well as multiple time steps. This has the advantage that temporal outliers, as well as bias coming from inaccurate intrinsic or extrinsic calibration of the cameras, gets filtered out.

To avoid sudden changes for the centroid and normal vector (which has a direct negative impact on the haptic sense of touch), a PID-based controller follows the target trajectory x and updates the final state $\bar{x}$ for each time step k:

$$\bar{x}^k = \text{PID}(\bar{x}^{k-1}, x^k), \quad x = \left[c_x^k, c_y^k, c_z^k, n_\theta^k, n_\phi^k\right]^T \tag{6}$$

Since the surface normal on average does not change with the same high rate as the centroid, the PID-values can be adapted accordingly, resulting in a smooth and realistic surface movement.

V. Haptic Force Rendering

In this final step, prioritized tasks for the hierarchical task controller are crafted using the resulting $\boldsymbol{c}^k$ and $\boldsymbol{n}^k$, and the HIP $\boldsymbol{h}^k$, that is a virtual point on the robot's body. We designed 3 haptic constraint types, namely a positional, velocity and force constraint.

The projected distance along the surface's normal direction can be calculated using the scalar product and subtracting the radius of the HIP:

$$d_{\text{proj}} = (\boldsymbol{c}^k - \boldsymbol{h}^k) \circ (-\boldsymbol{n}^k) - r_{\text{HIP}} \tag{7}$$

The corresponding constraint jacobian is given by projecting the translational spatial Jacobian for the HIP J_s to the surface normal vector n^k:

$$J_c = (n^k)^T \cdot J_{s,\text{trans}} \tag{8}$$

A. Positional Constraint

The positional inequality constraint is

$$\alpha_P^k < d_{proj} - k_p \Delta t \left(J_c \cdot \dot{q}\right) - \frac{1}{2} k_p^2 \Delta t^2 \left(\dot{J}_c \cdot \dot{q}\right) \tag{9}$$

$$J_{t,P} = \frac{1}{2} k_p^2 \Delta t^2 J_c \tag{10}$$

where k_p is the task space position constraint gain. The constraint is finally added to the task controller as derived by [14]:

$$J_{t,P} \cdot \xi = \alpha_P^k, \quad \xi = \begin{pmatrix} \ddot{q} \\ \tau \end{pmatrix} \tag{11}$$

The penetration depth is calculated by applying a 2nd order taylor expansion on the filtered joint velocities. $J_{t,P}$ is the task Jacobian that is given to the hierarchical task controller together with the constraint value α_P^k.

B. Velocity Constraint

For the velocity constraint, we defined a function that depends on the distance of the HIP to the collision point:

$$\mathbf{vc}(d) = \begin{cases} v_{\text{min}}, & \text{for } d_{proj} < d_{\text{offset}} \\ v_{\text{max}}, & \text{for } d_{proj} > d_{\text{offset}} + d_{\text{width}} \\ v_{\text{min}} + \frac{(d_{proj} - d_{\text{offset}})}{d_{\text{width}}} \cdot (v_{\text{max}} - v_{\text{min}}), & \text{else} \end{cases} \tag{12}$$

and the corresponding inequality value and task jacobian is given by

$$\alpha_V^k < \mathbf{vc}(d) - (J_c \cdot \dot{q}) - k_v \Delta t \left(\dot{J}_c \cdot \dot{q}\right) \tag{13}$$

$$J_{t,V} = k_v \Delta t J_c \tag{14}$$

This ensures a linear maximal speed towards the collision object, to prevent collisions with high velocities.

VI. Results & Discussion

Our collision avoidance pipeline is evaluated on ANYexo, a new 6-DoF torque controlled upper-limb exoskeleton, and two Intel Realsense D435 depth sensors. The hierarchical task controller is set to the following configuration for all experiments:

priority	task
1	Equation of Motion
2	Safety Limits for Joint Position and Velocities
3	GHB Fixed Collision Constraint
4	**Online Collision Avoidance**
5	Minimize Joint Accelerations

TABLE I

Hierarchical task controller configuration for the presented experiments.

A HIP is placed at the corner of the handle that faces the patient, and a radius of 0.04 m is chosen as an offset.

Fig. 8 shows the final surface movement compared to the planned trajectory and the raw estimations from the data processing pipeline (yellow scatter plot), as derived in Sec. IV-F. Note that raw data has a frequency of 30 Hz for each camera, while the resulting trajectory has a frequency of 800 Hz. Small bias coming from an inaccurate extrinsic

calibration of the cameras is also averaged with this method, assuming a mean bias of zero for multiple cameras.

We first perform an open-loop experiment to validate the

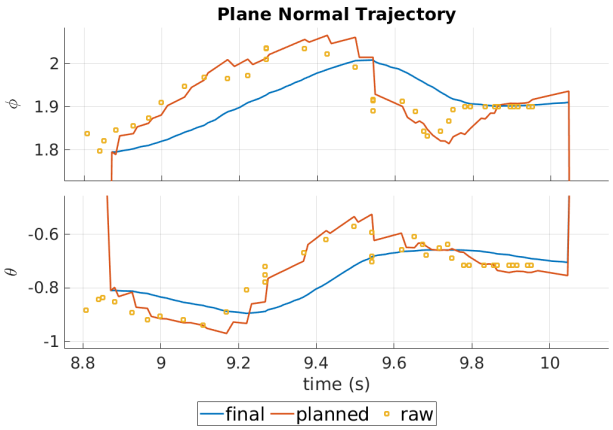

Fig. 8. This plot shows the raw normal vector estimations c_ϕ and c_θ in spherical coordinates, coming from the data processing pipeline at 30 Hz, the corresponding planned trajectory that is up-sampled to 800 Hz, and the final trajectory of the normal vector after applying PID control. For the centroid, this procedure is identical.

robustness of the collision avoidance pipeline and obstacle trajectory generation for multiple direct impacts. The collision centroid and normal vector are generated and visually validated, as seen in Fig. 9. As soon as the HIP has moved away far enough, the constraint gets disabled, as indicated by the gray areas.

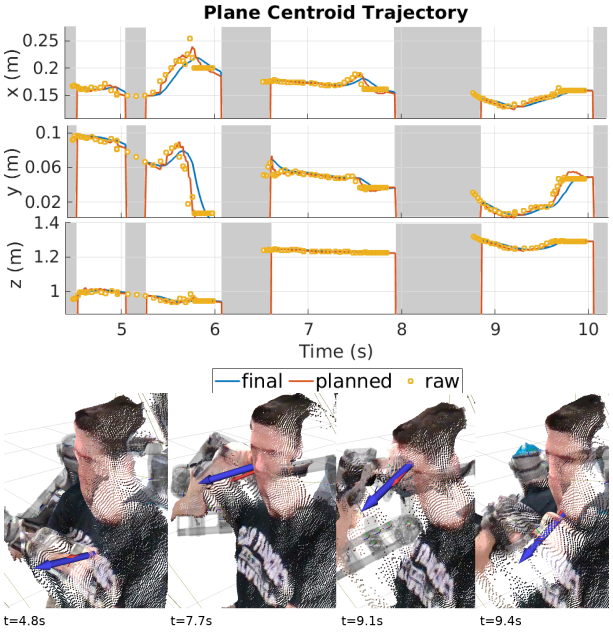

Fig. 9. Normal and centroid vector calculation of an example scene for multiple direct impacts with a healthy subject.

The second experiment is an open-loop experiment, to validate a correct input for the haptic rendering upon sliding along the head. Fig. 10 shows the resulting surface centroid and normal. For 6 time steps, the pictures show the current state of the point cloud as well as the surface vector markers (blue). The surface is rendered smoothly and no oscillations or sudden jumps in the surface estimation can be seen.

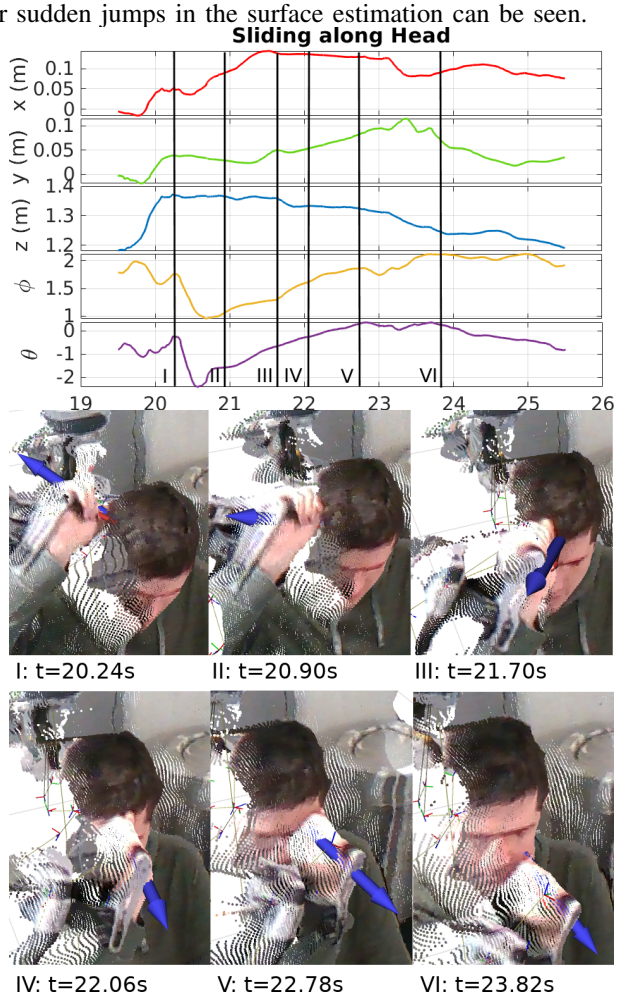

Fig. 10. Open-loop experiment for consecutive sliding along the head. The plot shows the surface centroid and normal. For 6 time steps, the point cloud state and surface vector markers are shown.

The closed-loop test looks at three consecutive direct impacts and sliding with enabled positional and velocity constraint. A virtual horizontal plane is at fixed distance to the floor. The robot end-effector is hit against this wall. Position and velocity is recorded. Fig. 11 shows the resulting relative position d_{proj} and velocity $\frac{\delta}{\delta t} d_{\text{proj}}$. For direct impact, upon hitting the virtual surface, the positional constraint acts as a spring, resulting in higher reaction forces for bigger penetration depths. Similar behaviour can be seen as slight oscillations in the sliding experiment. Also, note how the constraint influences the HIP velocity. A bouncing effect can be observed upon penetration of the surface. This can be explained by considering the reaction forces coming from the positional constraint; the HIP is pushed outwards, while the patient's inertia still acts in the opposite direction. The velocity boundary leads to the HIP hitting the surface a second time, before the arm is finally retracted.

The last experiment observes the behaviour of sudden movements of the collision surface. The surface is elevated with a constant speed. Resulting haptic collision point (HCP) and HIP positions are shown in Fig. 12 for different elevation

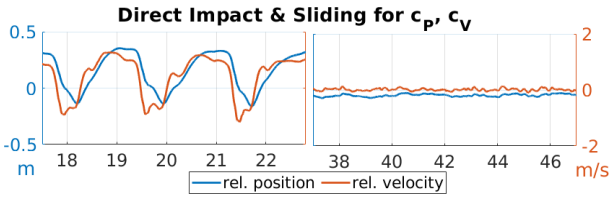

Fig. 11. Closed-loop one-dimensional experiment with enabled positional and velocity constraint. Note the small bouncing effect in the end-effector's relative velocity that arises from the patient's inertia acting in the opposite direction than the reaction forces.

speeds. It can be seen that the safety margin gets smaller for increased speeds, and the reaction is delayed. To counteract this behaviour, either the safety margin, that is the HIP's radius, can be increased, or the movement of the collision surface can be included in the velocity constraint and thus supporting the reaction from the positional constraint.

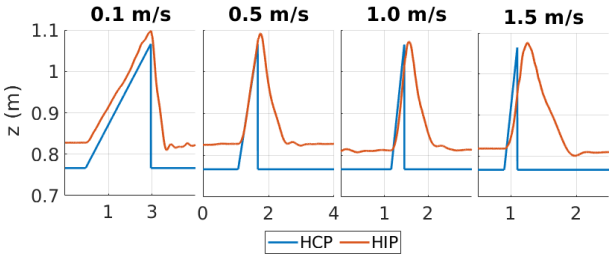

Fig. 12. Closed-loop one-dimensional experiment with moving horizontal surface at different speeds. Note how the safety margin gets smaller for increased speeds.

VII. Conclusion

The sequence of consecutive computer vision processes allow the estimation of closest collision surfaces for noisy and biased camera data. Occlusion problems with the exoskeleton robot are solved by introducing a novel surface reconstruction method. The positional and velocity constraint finally prevent collisions while allowing haptic interaction with the human body. Our method thus successfully closes the gap of enabling virtual haptic interaction with own unguided body parts in human-robot therapy.

Due to the framerate of the camera and the necessary trajectory generation step, relative speeds between the exoskeleton and the obstacle are limited to $1.0\ \mathrm{m/s}$ for a haptic point radius of $0.04\ \mathrm{m}$ with this method, and can thus be further improved.

The open-loop experiments show that our pipeline is robust to occlusion by the exoskeleton robot and has the potential to deal with high dynamic cases such as direct impacts, sliding along the unguided body parts, and sudden movements of the obstacle towards the haptic interaction point.

Although the one-dimensional closed-loop tests show promising results, additional closed-loop tests with complex surfaces need to be done in the future to assess the real potential of the method.

References

[1] K. B. Kaldestad, S. Haddadin, R. Belder, G. Hovland, and D. A. Anisi, "Collision avoidance with potential fields based on parallel processing of 3d-point cloud data on the gpu," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 3250–3257.

[2] G. Du, S. Long, F. Li, and X. Huang, "Active collision avoidance for human-robot interaction with ukf, expert system, and artificial potential field method," *Frontiers in Robotics and AI*, vol. 5, p. 125, 2018.

[3] A. M. Zanchettin, N. M. Ceriani, P. Rocco, H. Ding, and B. Matthias, "Safety in human-robot collaborative manufacturing environments: Metrics and control," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 882–893, April 2016.

[4] J. Minguez and L. Montano, "Extending collision avoidance methods to consider the vehicle shape, kinematics, and dynamics of a mobile robot," *IEEE Transactions on Robotics*, vol. 25, no. 2, pp. 367–381, April 2009.

[5] Y. Fu, G. Jiang, W. Feng, Y. Zhou, and Y. Ou, "On real-time obstacle avoidance using 3-d point clouds," in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, Dec 2014, pp. 631–636.

[6] S. Knopp, M. Lorenz, L. Pelliccia, and K. Philipp, "Using Industrial Robots as Haptic Devices for VR-Training," in *IEEE, Conference on Virtual Reality and 3D User Interfaces (VR)*, 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8446614

[7] Y. Zimmermann, A. Forino, R. Riener, and M. Hutter, "Anyexo: A versatile and dynamic upper-limb rehabilitation robot," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3649–3656, Oct 2019.

[8] H. Seraji and B. Bon, "Real-time collision avoidance for position-controlled manipulators," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 4, pp. 670–677, Aug 1999.

[9] A.Sharkawy, P. Koustoumpardis, and N. Aspragathos, "Human-robot collisoins detection for safe human-robot interaction using one multi-input-output neural network," *Soft Computing*, August 2019.

[10] K. Salisbury, F. Conti, and F. Barbagli, "Haptic Rendering: Introductory Concepts," *Computer Graphics and Applications*, vol. 1, no. 04, 2004.

[11] C. B. Zilles and J. K. Salisbury, "A constraint-based god-object method for haptic display," in *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, vol. 3, Aug 1995, pp. 146–151 vol.3.

[12] F. Rydén and H. J. Chizeck, "A proxy method for real-time 3-dof haptic rendering of streaming point cloud data," *IEEE Transactions on Haptics*, vol. 6, no. 3, pp. 257–267, July 2013.

[13] X. Li and T. Kesavadas, "Surgical robot with environment reconstruction and force feedback," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, July 2018, pp. 1861–1866.

[14] M. Hutter, H. Sommer, C. Gehring, M. Hoepflinger, M. Bloesch, and R. Siegwart, "Quadrupedal locomotion using hierarchical operational space control," *The International Journal of Robotics Research*, vol. 33, pp. 1047–1062, 07 2014.

[15] C. Dario Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, "Perception-less terrain adaptation through whole body control and hierarchical optimization," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots*, Nov 2016, pp. 558–564.

[16] A. Leeper, S. Chan, K. Hsiao, M. Ciocarlie, and K. Salisbury, "Constraint-based haptic rendering of point data for teleoperated robot grasping," in *2012 IEEE Haptics Symposium (HAPTICS)*, March 2012, pp. 377–383.

[17] K. Glass, R. Colbaugh, D. Lim, and H. Seraji, "Real-time collision avoidance for redundant manipulators," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 3, pp. 448–457, June 1995.

[18] ISO, *ISO/TS 15066:2017-04: Robots and robotic devices; collaborative robots*, Apr. 2017. [Online]. Available: https://www.iso.org/news/2016/03/Ref2057.html

[19] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[20] M. Korn, M. Holzkothen, and J. Pauli, "Color supported generalized-icp," in *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, vol. 3. Los Alamitos, CA, USA: IEEE Computer Society, jan 2014, pp. 592–599. [Online]. Available: https://doi.ieeecomputersociety.org/