Drawing With Operation Space Impedance Control

Chris Evagora Saim Naveed Iqbal Steve Carter Feujo Nomeny

***Abstract*—This project aims to implement an operation space impedance controller with inertia shaping on the KUKA iiwa. An operation space impedance controller imposes mass-spring-damper dynamics on the end effector, which is beneficial in tasks where you want to be in contact with the environment. A working impedance controller implemented, and tested by making the robot draw different trajectories on an uneven surface, without perception. We also test drawing in the case where we have perception, and see that this increases performance.**

I. Introduction

While the field of robotics strives to enable interactions with the real world, many implementations of industry robots still avoid contact with the environment. This is due to a number of reasons. For one, these robots can exert large forces, and if control is not implemented carefully they may damage the workspace, or in the worst case harm nearby operators or collaborators. Real-world environments are complex and unpredictable, making safe and efficient interactions with robots a difficult task. Therefore, one usually opts to design for a system that avoids contact entirely, removing the risk of exerting excessive force or encountering contact instability.

This is unfortunate as many tasks require contact such as painting, polishing, or sanding a surface. Contact can even enable nonprehensile manipulation which can allow us to solve much more sophisticated problems. A common solution to this problem is to introduce compliant behavior through what is called impedance control. In operation space, an impedance controller can impose virtual dynamics on the end effector, making it behave equivalently to mass-spring-damper-system. In addition to mitigating the risks associated with direct contact, impedance control allows following a trajectory without the use of inverse dynamics, by changing a virtual setpoint for the mass-spring-damper system.

To gain a deeper understanding of this control paradigm, we wish to implement OSC on the KUKA iiwa from scratch by directly controlling the torques applied at each joint. Through a force sensor placed before our end effector, we can even shape the inertia of the end effector during acceleration of the end effector due to external forces. The goal of OSC is to completely cancel the dynamics of the robot and replace it with desired dynamics, that of a 6-axis spring, mass, dashpot system for both orientation and position.

To test the effectiveness of our controller, we created a system that can draw an arbitrary trajectory on an unknown surface. Without the use of perception, we can use our orientation space pose error to trace curves on surfaces we do not know much about. By placing our desired position loosely beneath the surface which we want to draw on, we can rely on our operation space impedance control to drive the robot to the surface and apply a controllable force to it. Additionally, we add simple perception to our system, to test whether that will improve the performance.

II. Related work

Impedance control is a relatively old control paradigm, developed by researchers such as Neville Hogan in the 1980's [1]. Many real world implementations of force controllers do not program the experienced mass at the end effector, only the stiffness and damping, known as stiffness control. This is due to impedance control requiring expensive matrix inversions for inertia shaping as well as a high bandwidth force sensor at the end effector to appropriately scale external forces. Often, the added complexity does not justify the gain in performance. Our implementation began with the inspiration to fully implement OSC, including inertia shaping and force feedback using Drake's built-in multibody plant. On real hardware this would require a 6 DOF inline force plate which connects our robot to our end effector.

Even though drawing is a fairly niche subset of contact-based tasks, there has still been research on making robot arms draw. Some existing implementations use a hybrid force-position control [2] [3], which requires the surface to be known. Others have been able to achieve drawing on arbitrary surfaces using impedance control. We believe our implementation can operate in more general environments compared to previous work as our controller doesn't require any prior knowledge of the surface since we do not project the desired drawing trajectory on to the surface.

Additionally, we propose a method to integrate basic perception and use surface normal estimates of the drawing surface to increase performance. By doing so, our approach not only addresses the limitations of existing robotic control strategies but also enhances the robot's adaptability and accuracy in real-time, particularly in environments with complex or unknown geometries.

III. Approach

A. Simulation

We implement our solution in Drake, on the KUKA iiwa. A pen is welded to the last link of the iiwa, and a writing system is implemented. We design a bowl to act as our canvas, and add it to our scene. Finally we added 3 cameras, to capture a point cloud of our canvas. The complete system can be seen in 1. The most notable systems are the controller itself and the normal estimation, which we will explain more in depth. To cresimp

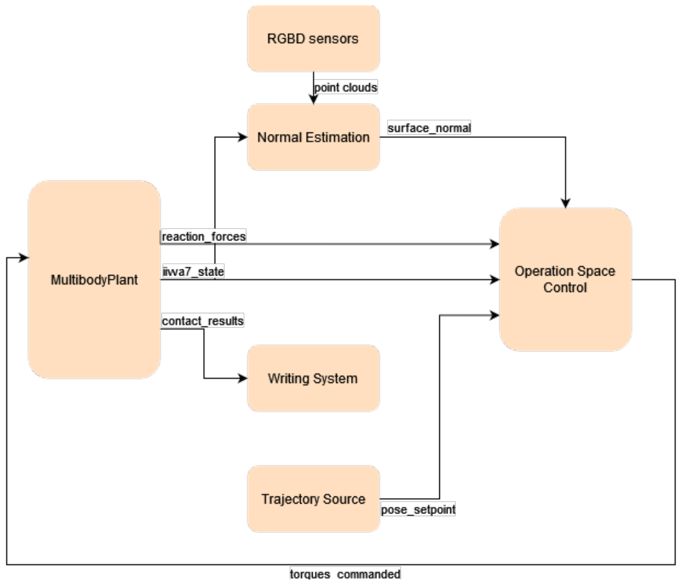

Fig. 1: System diagram

B. Operation Space Control

We begin with our desired dynamics in operation space, with desired mass, damping, and spring matrices M_{des}, B_{des}, and K_{des} respectively. Our operation space state variable includes both position and orientation, making each of the matrices above a 6x6 block matrix with the upper left 3x3 matrix for spatial moments and the lower right 3x3 matrix for spatial forces.

$$M_{des}\ddot{x} - B_{des}(\dot{x}_{des} - \dot{x}) - K_{des}(x_{des} - x) = F_{ext} \tag{1}$$

This is in juxtaposition to our actual robot dynamics, which is expressed in generalized coordinates, or joint space. We quantify the effects of torque applied at the motors and the presence of an external wrench at the end effector.

$$M_q\ddot{q} + V(q,\dot{q}) = \tau_{motor} + J^T F_{ext} \tag{2}$$

This can be rewritten in terms of commanded wrench F_{act} which can be realized as motor torques through multiplication with the Jacobian transpose of the end effector with the commanded wrench F_{act}.

$$M_q\ddot{q} + V(q,\dot{q}) = J^T(F_{act} + F_{ext}) \tag{3}$$

Isolate $\ddot{q}$ from (3).

$$\ddot{q} = -M_q^{-1}V(q,\dot{q}) + M_q^{-1}J^T(F_{act} + F_{ext}) \tag{4}$$

Recall the definition of the Jacobian of the end effector.

$$\dot{x} = J\dot{q} \tag{5}$$

Differentiate $\dot{x}$ to get acceleration in the task space.

$$\ddot{x} = \dot{J}\dot{q} + J\ddot{q} \tag{6}$$

Plug (4) into (6) to get $\ddot{x}$, the acceleration in the task space given commanded wrench F_{act}.

$$\ddot{x} = JM_q^{-1}J^T(F_{act} + F_{ext}) - JM_q^{-1}V(q,\dot{q}) + \dot{J}\dot{q} \tag{7}$$

Define M_x to simplify future expressions.

$$M_x = (JM_q^{-1}J^T)^{-1} \tag{8}$$

Rearrage equation (7) with M_x.

$$M_x\ddot{x} + M_xJM_q^{-1}V(q,\dot{q}) - M_x\dot{J}\dot{q} = F_{act} + F_{ext} \tag{9}$$

Recall desired dynamics (1), isolate $\ddot{x}$ and insert into (9).

$$\begin{aligned}M_xM_{des}^{-1}(B_{des}(\dot{x}_{des} - \dot{x}) + K_{des}(x_{des} - x) + F_{ext}) +& \\ M_xJM_q^{-1}V(q,\dot{q}) - M_x\dot{J}\dot{q} - F_{ext} = F_{act}&\end{aligned} \tag{10}$$

Combine like terms and find F_{act}, the required wrench at the end effector in order to impose virtual dynamics from (1).

$$\begin{aligned}F_{act} = M_xM_{des}^{-1}(B_{des}(\dot{x}_{des} - \dot{x}) + K_{des}(x_{des} - x)) +& \\ M_xJM_q^{-1}V(q,\dot{q}) - M_x\dot{J}\dot{q} + (M_xM_{des}^{-1} - I)F_{ext}&\end{aligned} \tag{11}$$

Now find joint torques that give F_{act} at the end effector. This is sent to the robot to achieve OSC.

$$\tau_{motor} = J^T F_{act} \tag{12}$$

With our implementation, finding the force F_{ext} on the end effector proved to be more complex and not as useful as originally thought. In OSC, measuring the external wrench on the end effector allows for amplification of that wrench in the directions of higher robot inertia and attenuation in directions of lower robot inertia such that the overall inertia of the end effector appears as desired in the imposed virtual dynamics. In the case of drawing on a nearly flat surface, the effects of the robot's inertia in different directions does not affect much the operation of drawing since the acceleration of the end effector while tracking is very small. Originally, F_{ext} was going to be measured from the "reaction_forces" output of the iiwa multibody plant. This did not quite work as planned because spatial forces are still exerted on the end effector despite the absence of contact since gravity always acts on all bodies. In the derivation of OSC above, gravity is not accounted or compensated for. This can be supplemented with a parallel gravity compensation controller which sums its commanded joint torques with the OSC, except for the fact that the multibody plant will still measure a reaction force at the end effector due to gravity. It was therefore decided to not include force feedback into this implementation of OSC since it did not help with performance.

In order to track the desired trajectory on the unknown surface, a rough point cloud of the surface is captured with depth imaging cameras, downsampled, and processed to find surface normals for the *k* nearest neighbors to the end effector. The end effector as well as the virtual spring mass damper system parameters are rotated to meet with the surface perpendicularly. Upon coming into contact with the surface as shown in Figure 2, the controller switches from trying to track the desired pose in the trajectory to tracking the orientation of the surface normal as well as the closest point on the Z-axis of the desired pose in the trajectory, which in our case is usually aligned with the world Z-axis.

The translational error vector $\vec{e}$ shown in Figure 2, given the position of the end effector $\vec{p_e}$, the position of the desired pose $\vec{p_d}$, and the unit length z-axis of the desired pose $\vec{z}$, can be calculated as:

$$\vec{e} = (\vec{z} \times (\vec{p_d} - \vec{p_e})) \times \vec{z} \tag{13}$$

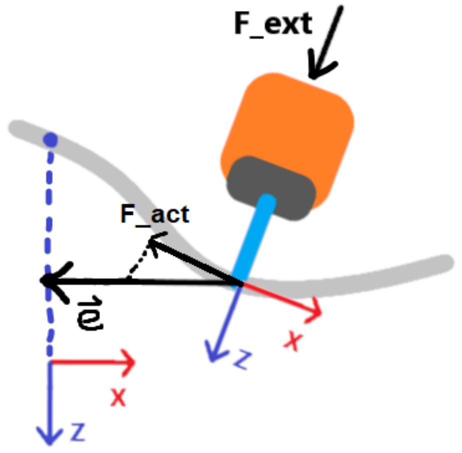

Fig. 2: Commanded actuated force and virtual external force when in contact. Commanded force F_act is the translational error defined in equation (13), scaled and projected onto the plane of contact. Z-axis of the end effector is driven to be coaxial with surface normal.

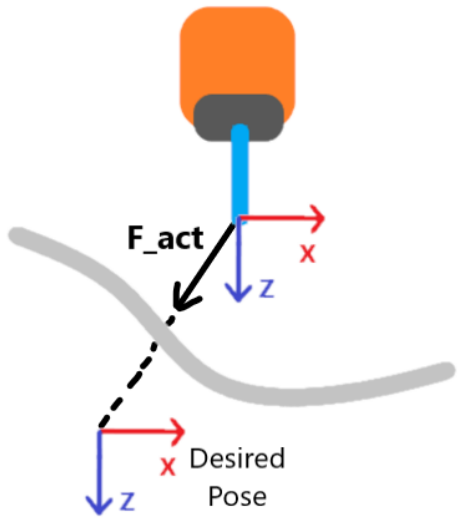

Fig. 3: Commanded actuated force when not in contact. This controller mode is also used in contact when there is no perception since contact cannot be detected well. Orientation of desired pose is also tracked.

This error vector is passed as the position error to the OSC. The gains for the translational spring, mass, dashpot system are chosen such that the spring constant in the Z-axis of the end effector while in contact is set to zero. The force that the error vector produces when multiplied by the desired stiffness matrix, K_d, therefore projects the error vector into the plane normal to the surface where the end effector is in contact. This ensures that all corrective forces, F_{act}, tracking the projection of the trajectory along the Z-axis of the desired pose act only tangentially to the surface.

Contact is detected when our end effector is calculated to be within a certain distance from the nearest point in the cloud. While in contact, a virtual force is injected into F_{ext} to make the robot push into the surface with a constant force. Alongside this, the spring constant aligned with the Z-axis in task space is set to zero to allow F_{ext} to push the end effector into the surface.

C. *Jacobian Null Space Controller*

Since we are commanding torques to control the pose of the end effector in space, we require 6 DOF to be fully actuated. The IIWA robot by KUKA is a 7 DOF robot arm that must have its last DOF controlled to prevent potential stability or kinematic problems. For this a null space position controller was added to fill the gaps of the OSC. Using the dynamically consistent generalized psuedo inverse of the Jacobian of the end effector, described in detail by studywolf [5], we can constrain the last DOF. We define a nominal robot configuration and apply a PID controller to command joint torques to reach this configuration:

$$q_0 = [0, 0, 0, 0, 0, 0, 0] \tag{14}$$

$$\tau_{motor} = Kp \cdot (q_0 - q) + Kd \cdot (-\dot{q}) \tag{15}$$

We then calculate the psuedo inverse of the Jacobian transpose and use it to project the desired torque vector of the PID controller into the null space of the Jacobian.

$$J^{T+} = M_x J M_q^{-1} \tag{16}$$

$$\tau_{motor,projected} = (I - (J^T J^{T+}))\tau_{motor} \tag{17}$$

The output of the projection is simply summed with the output of the OSC, as well as the gravity compensation controller trivially implemented in drake, to produce the final output torque vector sent to the IIWA.

D. *Normal Estimation*

In the normal estimation step, the objective is to compute the surface normal for each point in the point cloud. This is achieved through the following steps:

1) **Nearest Neighbors Search**: For a selected query point q in the point cloud, this is the tip of the pen in our case, a k-d tree is used to find its k nearest neighbors, forming a local neighborhood $N(q)$.

$$N(q) = \text{KDTree(PointCloud).query}(q, k) \tag{18}$$

2) **Principal Component Analysis (PCA)**: Apply PCA on the points in $N(q)$ to compute the covariance matrix. The covariance matrix is then decomposed to find its eigenvalues and eigenvectors.

$$\text{Cov}(N(q)) = \frac{1}{k} \sum_{p \in N(q)} (p - \mu)(p - \mu)^T \tag{19}$$

where μ is the mean of the points in $N(q)$.

3) **Normal Vector Determination**: The normal vector at q is identified as the eigenvector corresponding to the smallest eigenvalue, as it represents the least variance direction in the local neighborhood.
4) **Orientation Adjustment**: Ensure that the normal vector is oriented towards the camera. If the dot product of the normal vector with the vector from the point to the camera position is negative, the normal vector is inverted.

Following these steps we can accurately estimate the normals of the surface represented by the point cloud.

IV. Results

We test the fully implemented system with a two different trajectories, one resembling a sine wave, the other in the shape of an ”M”. The controller is tested in both the case where we have perception, and the case when we don’t. The performance of the controller is rated qualitatively, by looking at the quality of the drawing.

Figure 4 shows the resulting drawing when the trajectory reference is a sinusoidal wave, and the normal estimation is used. We see that the robot manages to track the trajectory fairly well, even at the steepest parts of the surface. We see a similarly good performance when testing other trajectories. When turning off the rotation of forces and end effector, and only relying on the stiffness of the system to track the surface, we see that the quality of the drawing is reduced. This is clear in figure 6. We see that the we have less consistent contact with the surface, in addition to what appears to be some slight distorion caused by the surface.

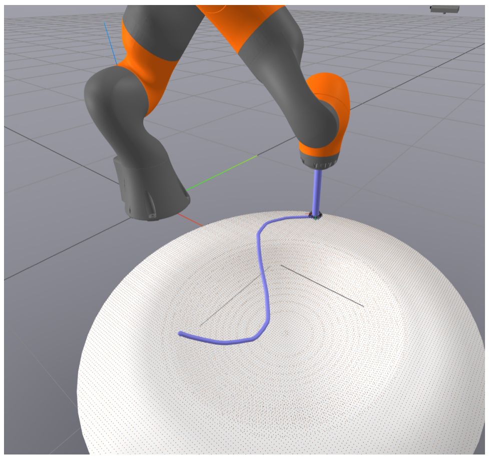
Fig. 4: Resulting drawing when end effector and virtual dynamics parameters are rotated perpendicular to surface. Coarseness of sinusoid is due to underlying trajectory only having 10 setpoints along trajectory.

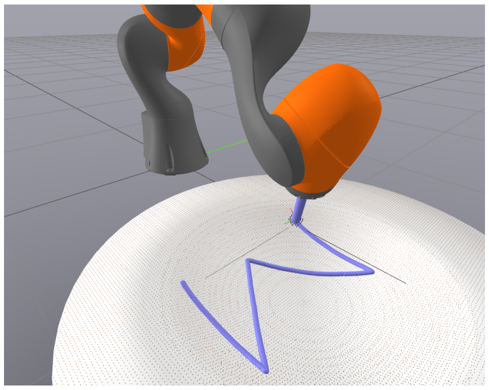
Fig. 5: Resulting drawing for an M-shaped trajectory for rotated parameters.

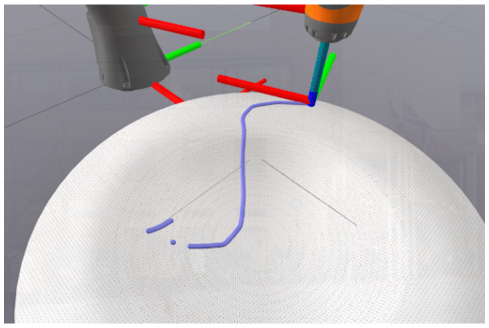
Fig. 6: Resulting drawing without rotating end effector or virtual dynamics parameters.

V. Discussion

Our solution gives us the desired result: a robot that is able to draw on a surface without needing any knowledge about it. This shows that impedance control is quite powerful tool in enabling control. We believe our implementation could be expanded to other use cases, such as engraving, cutting, polishing or other tasks where you want to follow a path along a surface. We do however see that the completely blind result isn’t perfect, as it is difficult to maintain contact at all points along the surface.

Adding perception and using it to rotate the stiffnesses makes the drawing demonstrabely better. However, as we compute normals in each time step, it impacts performance greatly. Of course this can be optimized, but there will still be a tradeoff between computational load and the quality of the controller. Whether or not the sacrifice in runtime is worth it will be dependant on application, as in some cases one may prefer perfect tracking, or perhaps one would like to avoid exerting forces in non-tangential directions.

The simulation environment was established with three cameras, each capturing a snapshot of the scene and recording the surface’s point cloud (excluding the iwaa), for subsequent runtime usage. Since the environment is static, we do not need continuous capture. In scenarios where the surface is subject to dynamic changes, continuous point cloud acquisition is imperative. This allows for real-time adjustments in normal estimation and trajectory planning, accommodating the dynamic nature of the environment. While this adds computational and logical complexity, it significantly enhances the system's robustness.

Finally, our controller was only tested on one surface, and on two relatively simple shapes. As such, we do not truly know the limits of our implementation. Much time went into developing the controller itself, if we had more time we would have liked to test it with more complex surfaces or trajectories. We imagine our controller would struggle with less smooth surfaces, or surfaces with very steep slopes. Still, we are very satisfied with the performance.

VI. Conclusion

While our controller lacked force feedback for proper inertia shaping during acceleration caused by external forces, we have still showed that this method is a powerful tool in enabling force control of the end effector. Our solution is able to draw arbitrary paths on arbitrary surfaces, even when we have no perception of the surface. Adding perception improves the tracking of the trajectory, but at the cost of slower computation. Our blind controller solution also suffers from some distortion caused by the surface since we don't explicitly track the projection of the trajectory on the surface. Nevertheless, we believe our implementation performs well, and can be expanded to other applications, such as engraving or cutting.

Contributions

The team did it's best to assign tasks equally, and help eachother out when needed. Chris spent a significant effort on implementing the impedance controller in Drake. Saim helped implement rotation of the end effector and rotation of the virtual dynamics parameters with input surface data. Steve implemented the simulation setup and the normal estimation, and Saim implemented the writing system as well as creating the drawing surface.

References

[1] N. Hogan, "Stable execution of contact tasks using impedance control," *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, pp. 1047–1054, 1987.

[2] T. Weingartshofer, A. Haddadi, C. Hartl-Nesic, and A. Kugi, "Flexible Robotic Drawing on 3D Objects with an Industrial Robot," *2022 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 29-36, 2022.

[3] A. Haddadi, "Robotic drawing on a 3D object," Diploma Thesis, Wien, 2020.

[4] D. Song, T. Lee and Y. J. Kim, "Artistic Pen Drawing on an Arbitrary Surface Using an Impedance-Controlled Robot" *IEEE International Conference on Robotics and Automation*, pp. 4085-4090, 2018.

[5] StudyWolf Blog, "Robot Control 5: Controlling in the Null Space," https://studywolf.wordpress.com/2013/09/17/robot-control-5-controlling-in-the-null-space/