Robot Art Competition Using HERB: Final Report

Ryan Gibbs (rgibbs), Dorothy Kirlew (dkirlew), Astha Prasad (asthap), Sida Wang (sidaw)

1. Problem Description

The purpose of this project is for a user to teleoperate HERB using a Razer Hydra to create 6 paintings that will be submitted to the RobotArt International Competition. The user will use a binary toggle to lock and unlock HERB's arm to the plane of the canvas. The user will use a second binary toggle to rotate HERB's wrist between the straight, painting position and the bent, dipping position to apply paint to the brush. The user will hold down the analog trigger to make short, controlled movements with HERB's arm. These features can be combined to dip HERB's wrist in order to apply paint to the attached paintbrush, straighten HERB's wrist to the painting position, servo to the canvas, lock the paint brush to the plane of the canvas to ensure the paintbrush stays on the canvas, apply several strokes of paint, and unlock from the plane in order to reposition the paint brush or apply more paint.

Real-time control of end effectors with constraints is a difficult task. Inverse kinematic planners are also difficult. Some artists may not be strong or steady enough to paint or continue painting due to Parkinson's or Cerebral Palsy. Using HERB as a paintbrush would take the strain off the user, as well as give them smoother control over their art.

2. Implementation

a. IK via inverted Jacobian

In order to intuitively teleoperate Herb by having the end-effector mimic the movements of the teleoperator, the inverse kinematics problem needed to be solved. Solving this would allow the relative movements of the Razer Hydra to be converted into the joint angle changes that need to be made in HERB's arm. Luckily, HERB has the ability to compute the Jacobian for each arm on demand. The Jacobian encodes how small changes in each joint angle affect the position and orientation of the end effector. By inverting the Jacobian, how the changes in end effector orientation and position affect the joint angles can be determined using the forward kinematics formula (1) and inverse kinematics formula (2),

$$p(x_1) \approx p(x_0) + J_p(x_0)\Delta x \tag{1}$$

$$\Delta x \approx J_p^+(x_0)\Delta p, \tag{2}$$

where x represents joint positions and p represents the homogeneous pose of the end effector.

By setting the change in the orientation component of the homogeneous pose to zero, the computed solution to the inverse kinematics problem produces a result that maintains the current orientation of the end effector. This is critical to painting on the canvas and ensuring simple teleoperation control of the tip of the paintbrush.

b. Razer Hydra

The Razer Hydra is primarily a motion and orientation detecting game controller. This controller is used to obtain the following data:

1) Relative 3D position coordinates as floats
2) Toggle indicating “locked-to-plane” as a boolean
3) Teleoperation usage as a boolean dead man’s switch
4) Toggle to “dip” or “un-dip”

The communication between the Razer Hydra and HERB is done using a ROS custom message that publishes the above data.

There is a significant amount of noise in the initial raw data read from the Hydra. This incoming data is smoothed using a linear smoothing algorithm that averages the 15 most recent data values. The difference between the last averaged position and the current averaged position is then used as the 3D relative position of the Hydra. Because the rate at which the data is being published is very high, the relative position is too small to be useful to HERB. Therefore, data is sampled, smoothed, and published every 10 data points. The received data is then scaled by 5 to create the movements on a realistic scale.

c. Lock-to-Plane

The lock-to-plane functionality enables the user to lock HERB’s end effector to a plane that is parallel to the canvas introduced in HERB’s environment. It is implemented by a simple button press on the controller that operates as a toggle switch. Locking to a plane ensures that the paintbrush remains in contact with the canvas while the user is painting. When locked to the plane, any movements made in the 3-dimensional world are projected onto a 2-dimensional plane, thus rejecting any 3D movements that would pull the brush away from the canvas or push the brush through it.

The projection of a point onto a 2D plane is obtained by using a vector normal to the canvas and a point on the canvas. Both of these were obtained while inserting the canvas into the robot’s environment. The projected point, Q projected is obtained using equation 3,

$$Q_{projected} = Q - dot(Q - P, n) * n \tag{3}$$

where Q is the point in 3 dimensions, P is a point on the canvas, and n is the unit normal vector, as seen in Figure 1.

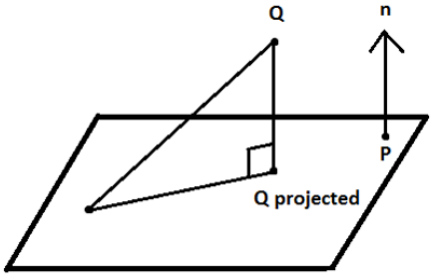

Figure 1: Projected of Point onto Plane

d. Look-Ahead Collision Checker

In addition to a simple collision checker between HERB, the canvas, and the table, a look-ahead collision checker was implemented. When the relative X, Y, and Z positions are received over a custom ROS message, the look ahead computes the Jacobian, not just for one movement, but if that movement is projected forward *N* times. If in any of these *N* movements, there is a collision between HERB, the canvas, and the table, no movement is made with HERB until the user releases the analog trigger and then re-engages the trigger. By doing so, it informs the user that their move is too strong or in the wrong direction and they must change what they are doing in order to continue teleoperating HERB. After several tests, it was determined that 2 look-aheads were sufficient to enable the user to teleoperate as needed while ensuring that there will be no collisions.

The look-ahead checker also serves to ensure that the user moves more slowly when it is approaching an obstacle. Small movements allow the user to get very close to an obstacle, as several small movements are unlikely to cause a collision. However, large movements are not allowed because several large movements would cause a collision. This enables the user to get the paintbrush very close to the canvas and adjust the amount of contact the brush has to create a fine or a bold stroke.

e. Orientation Change

The system's implementation of inverse kinematics preserves orientation, requiring a different way to change the orientation of the paintbrush was needed so that it could be dipped in paint or water. Since this action is performed infrequently, HERB's onboard planners were used to compute and execute a trajectory that toggled the orientation of the paintbrush between downwards or outwards. It is important to note that due to joint limitations and the restriction that only HERB's hand and wrist should move, the angle of the dipping position should not be straight down, but rather at some small angle to avoid the singularity at the wrist. Similarly, the angle of HERB's hand when holding the paintbrush outwards to paint is not purely horizontal. This is to bias the wrist position in order to make future orientation changes simpler.

3. Results

Using the methods described above, users were able to successfully teleoperate HERB to paint several pieces (Figure 2 through Figure 7). HERB moved at a rate almost identical to that of the teleoperator. At no point did HERB collide with itself or any part of the environment due to both the intelligence of the user and the look-ahead collision checker.

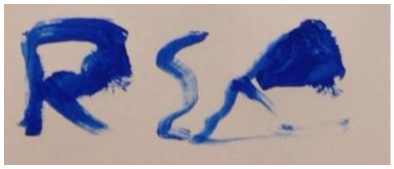

Figure 2: R Σ Triangle

Figure 3: Happy Sea Demon

Figure 4: Angler Fish Attack on Love

Figure 5: Swan

Figure 6: Red Broccoli

Figure 7: Woman on Trapeze Fleeing Sun

a. Initial Work

Initially, the Herbpy method “PlanToEndEffectorOffset” was used to control Herb’s motions. However, the time required for this function to execute, even for relatively short trajectories, was still on the order of seconds, rather than milliseconds. This prompted an attempt at throttling the publish rate of the Razer Hydra to compensate and providing fewer commands that needed to be computed. Unfortunately, this also proved ineffective, as it resulted in a loss of interactivity and a large delay in motions of the Razer Hydra and the execution of those motions by HERB.

Another facet that was attempted and discarded was to have HERB’s end effector snap to the plane. However, this feature was likely to involve a large planning component and computation time if the button was toggled far from the canvas. Furthermore, it removed all granularity from the user in terms of determining the distance between the paintbrush and the

canvas. Initial testing quickly showed that this distance had a direct effect on stroke width, allowing for bolder, wider strokes if the tip of the paintbrush was close to the canvas and thinner, more delicate strokes if the tip was further. Allowing the user to teleoperate to the desired distance and then merely constrain motion to a parallel plane, preserved the ability to choose stroke width and allow the user to maintain precise control of their art.

b. Conclusion

There are a few instances found during testing using simulation that were different than testing in real life. Even though the legs of the easel were taped down, it would still slowly slide away, causing the paintbrush and canvas to no longer be in contact. Additionally, it became much clearer how different HERBs movements are from that of a human, particularly around the elbow joint.

Throughout testing and use of HERB, few areas that could be improved upon where noted. The simplest improvements could be making the simulated environment more closely match the real environment by creating a more accurately sized and located canvas, and adding the paint cups to the table. Another improvement could be to restrict HERB's movement as the arm approaches a singularity in a manner similar to how HERB's movements are restricted when it approaches a collision. This would allow the user to spend more time painting and less time attempting to manipulate HERB out of a complicated configuration.

A video of the work done with HERB can be found here.