**S2 Appendix - Derivative-Free Optimization Algorithm**

1. **Preliminary Concepts**
	1. **Inverse Kinematics**

This section reviews preliminary concepts on creating a joint space trajectory from an operational space trajectory [1]. These concepts should be well-known to readers with a robotics background but could be unfamiliar to readers in the field of biomechanics.

Let ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \eev \in \reals{6} \] \end{document}]() denote the vector of the concatenated end-effector linear and angular velocity vectors:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[  \eev = \begin{bmatrix}  \posv \\ \angv  \end{bmatrix} \] \end{document} |  | (S2.) |

Here, ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \posv \in \reals{3} \] \end{document}]() denotes the end-effector linear velocity and ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \angv \in \reals{3} \] \end{document}]() denotes the end-effector angular velocity. Likewise, let ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rj \in \reals{n} \] \end{document}]() denote the vector of robot joint angles and ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rjv \in \reals{n} \] \end{document}]() denote the vector of robot joint velocities, where ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ n \] \end{document}]() represents the number of degrees of freedom of the robotic manipulator. For the robotic manipulator used in this study, ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ n=6 \] \end{document}]().

Then the differential kinematics equation:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[  \eev = \J(\rj)\rjv \] \end{document} |  | (S2.) |

provides a linear mapping from the robot joint velocities to the end-effector velocity in terms of the geometric Jacobian, ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \J \] \end{document}]() [1]. This relationship can be inverted to obtain joint velocities from the end-effector velocity:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rjv = \Jinv(\rj) \eev \] \end{document} |  | (S2.) |

The geometric Jacobian of the robotic manipulator in this study is a square matrix so it can be inverted when non-singular. Solutions for dealing with non-square Jacobian matrices exist but they are beyond the scope of this manuscript [1]. To deal with singularities the Jacobian pseudo-inverse (![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \Jpinv \] \end{document}]()) is utilized, which solves Equation (S2.3) in the least-squares sense [2]. Near a singularity, both the Jacobian inverse and pseudo-inverse lead to very high joint velocities. A variety of techniques (e.g. damped least-squares [2]) exist to address this issue but these are not necessary as part of the presented algorithm because high joint velocities are penalized via a cost function.

If the initial manipulator posture ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rj(0) \] \end{document}]() is known, joint angles can be computed by integrating their velocities over time [1]:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rj(t) = \int_{0}^{t} \rjv(\tau) d\tau + \rj(0) \] \end{document} |  | (S2.) |

The integration above can be performed in discrete time by utilizing numerical techniques such as the Euler integration method (![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \varDelta t = \tk{k+1}-\tk{k} \] \end{document}]()) [1]:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rjt{k+1} = \rjt{k} + \rjvt{k} \varDelta t \] \end{document} |  | (S2.) |

Combining equations (S2.3) and (S2.5), but using the Jacobian pseudo-inverse (![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \Jpinv \] \end{document}]()) instead of the Jacobian inverse (![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \Jinv \] \end{document}]()) leads to:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rjt{k+1} = \rjt{k} + \Jpinv(\rjt{k}) \eevt{k} \varDelta t \] \end{document} |  | (S2.) |

The Euler method can be derived from the forward finite difference formula, which is used to approximate ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \eevt{k} \] \end{document}]():

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \eevt{k} = \frac{\eet{k+1} - \eet{k}}{\varDelta t} \] \end{document} |  | (S2.) |

Combining equations (S2.6) and (S2.7) leads to a recursive formula for obtaining a joint space trajectory, ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rjt{k} \] \end{document}](), from an end-effector trajectory, ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \eet{k} \] \end{document}](), given an initial manipulator posture, ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rjt{0} \] \end{document}]():

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rjt{k+1} = \rjt{k} + \Jpinv(\rjt{k}) (\eet{k+1} - \eet{k}) \] \end{document} |  | (S2.) |

* 1. **Orientation Error**

Although it is evident how to compute ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \eet{k+1} - \eet{k} \] \end{document}]() for position, a method for computing ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \eet{k+1} - \eet{k} \] \end{document}]() for orientation must be established. Since Equation (S2.8) is the discretized version of Equation (S2.3) the orientation components of ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ (\eet{k+1} - \eet{k})/\varDelta t \] \end{document}]() should equal ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \angv \] \end{document}]() as ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \varDelta t \to 0 \] \end{document}]().

Let ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rotm{R}{EE}(\tau) \] \end{document}]() be the rotation matrix trajectory representing the orientation of the end-effector (EE) with respect to the base of the robot (R). Let ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \angv[R](\rotm{R}{EE}(\tau), \tk{k}) \] \end{document}]() be the angular velocity vector of ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rotm{R}{EE}(\tau) \] \end{document}]() evaluated at time ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \tk{k} \] \end{document}](). Likewise, let ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rotv[R](\rotm{R}{EE}(\tau), \tk{k}) \] \end{document}]() be the rotation vector [3] of ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rotm{R}{EE}(\tau) \] \end{document}]() evaluated at time ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \tk{k} \] \end{document}](). Then ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \eet{k+1} - \eet{k} \] \end{document}]() should be computed as:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \eet{k+1} - \eet{k} = \begin{bmatrix} \post[R]{k+1} - \post[R]{k} \\ \rotv[R](\rotm{R}{EE}(\tau) \cdot \rotm{R}{EE}^T(\tk{k}), \tk{k+1}) \end{bmatrix} \] \end{document} |  | (S2.) |

In general, the rotation vector velocity - ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rotvv[R](\rotm{R}{EE}(\tau), \tk{k}) \] \end{document}]() - is not equal to the angular velocity vector, but they are related by the relationship in Equation (S2.10) [4].

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[  \begin{split}  \angv[R](\rotm{R}{EE}(\tau), \tk{k}) &= \rotvv[R](\rotm{R}{EE}(\tau), \tk{k}) + \frac{1}{2!}(\rotv[R](\rotm{R}{EE}(\tau), \tk{k}) \times \rotvv[R](\rotm{R}{EE}(\tau), \tk{k}))\\ &+ \frac{1}{3!}(\rotv[R](\rotm{R}{EE}(\tau), \tk{k}) \times (\rotv[R](\rotm{R}{EE}(\tau), \tk{k}) \times \rotvv[R](\rotm{R}{EE}(\tau), \tk{k}))) + ...  \end{split} \] \end{document} |  | (S2.) |

Now let the first argument of ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \angv \] \end{document}](), ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rotv \] \end{document}](), and ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rotvv \] \end{document}]() be ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ {\rotm{R}{EE}(\tau) \cdot \rotm{R}{EE}^T(\tk{k})} \] \end{document}](). Note that![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rotv[R](\rotm{R}{EE}(\tau) \cdot \rotm{R}{EE}^T(\tk{k}), \tk{k})=\vect{0} \] \end{document}](). Hence Equation (S2.10) becomes:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[  \begin{aligned}  \angv[R](\rotm{R}{EE}(\tau) \cdot \rotm{R}{EE}^T(\tk{k}), \tk{k}) &= \rotvv[R](\rotm{R}{EE}(\tau) \cdot \rotm{R}{EE}^T(\tk{k}), \tk{k}) \\  \angv[R](\rotm{R}{EE}(\tau), \tk{k}) &= \rotvv[R](\rotm{R}{EE}(\tau) \cdot \rotm{R}{EE}^T(\tk{k}), \tk{k})  \end{aligned} \] \end{document} |  | (S2.) |

The simplification of Equation (S2.11) is proved at the end of this Appendix in Lemma 1.

Equation (S2.12) computes ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rotvv[R](\rotm{R}{EE}(\tau) \cdot \rotm{R}{EE}^T(\tk{k}), \tk{k}) \] \end{document}](), again, letting ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \varDelta t = \tk{k+1}-\tk{k} \] \end{document}]():

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \begin{aligned} \rotvv[R](\rotm{R}{EE}(\tau) \cdot \rotm{R}{EE}^T(\tk{k}), \tk{k}) &= \lim_{\varDelta t \to 0} \frac{\rotv[R](\rotm{R}{EE}(\tau) \cdot \rotm{R}{EE}^T(\tk{k}), \tk{k+1}) - \rotv[R](\rotm{R}{EE}(\tau) \cdot \rotm{R}{EE}^T(\tk{k}), \tk{k})}{\varDelta t} \\ &= \lim_{\varDelta t \to 0} \frac{\rotv[R](\rotm{R}{EE}(\tau) \cdot \rotm{R}{EE}^T(\tk{k}), \tk{k+1}) - \vect{0}}{\varDelta t} \\ &= \lim_{\varDelta t \to 0} \frac{\rotv[R](\rotm{R}{EE}(\tau) \cdot \rotm{R}{EE}^T(\tk{k}), \tk{k+1})}{\varDelta t} \end{aligned} \] \end{document} |  | (S2.) |

Hence, in the limit ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \varDelta t \to 0 \] \end{document}]() Equation (S2.12) computes ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \angv[R](\rotm{R}{EE}(\tau), \tk{k}) \] \end{document}](), justifying Equation (S2.9).

It's insightful to note that:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rotm{R}{EE}(\tk{k+1}) \cdot \rotm{R}{EE}^T(\tk{k}) = \rotm{R}{EE}(\tk{k}) \cdot \rotm{R}{EE}^T(\tk{k}) \cdot \rotm{R}{EE}(\tk{k+1}) \cdot \rotm{R}{EE}^T(\tk{k}) \] \end{document} |  | (S2.) |

Equation (S2.13) demonstrates that ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rotm{R}{EE}(\tk{k+1}) \cdot \rotm{R}{EE}^T(\tk{k}) \] \end{document}]() represents the rotation of the end-effector from time ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \tk{k} \] \end{document}]() to ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \tk{k+1} \] \end{document}]() in the reference frame of the robot (R).

* 1. **Joint Space and Operational Space Trajectory Computations**

Equation (S2.8) in Section 1.1 proposes a recursive formula for obtaining a joint space trajectory from an end-effector trajectory, given an initial manipulator posture. In this section, detailed steps for implementing this recursive procedure are provided.

The motion capture trajectories in this study are comprised of the pose of the humerus (H) measured in the motion capture (MC) laboratory frame of reference, encompassed in a 4x4 homogeneous transformation matrix, at equally spaced intervals of time: ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \homtt{MC}{H}{0} \] \end{document}](), ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \homtt{MC}{H}{1} \] \end{document}](),...,![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \homtt{MC}{H}{m-1} \] \end{document}]() - where ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ m \] \end{document}]() denotes the number of timepoints. Since the rigid-body relationship between the end-effector (EE) and the humerus (H) attached to the robot is known, ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \homt{EE}{H} \] \end{document}](), then the pose of the end-effector in the motion capture reference frame is:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \homtt{MC}{EE}{k} = \homtt{MC}{H}{k} (\homt{EE}{H})^{-1} \] \end{document} |  | (S2.) |

Given an initial manipulator posture ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rjt{0} \] \end{document}](), the forward kinematics function of the robot, ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ K \] \end{document}](), is utilized to compute the initial pose of the end-effector [1]:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \homtt{R}{EE}{0} = K(\rjvt{0}) \] \end{document} |  | (S2.) |

The pose of the end-effector at each subsequent timepoint can be computed from its rigid-body relationship with the initial end-effector pose:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \homttd{R}{EE}{k} = \homtt{R}{EE}{0} (\homtt{MC}{EE}{0})^{-1} (\homtt{MC}{EE}{k}) \] \end{document} |  | (S2.) |

The use of a superscripted 'D' denotes that this is the desired operational space trajectory for which we seek a suitable joint space trajectory. Several factors, such as geometric Jacobian singularities and drifting away from the solution due to linearization, can prevent finding a joint space trajectory that corresponds to the desired operational space trajectory [1]. The operational space trajectory that corresponds to the joint space trajectory will be called the *achieved operational space trajectory* and will be denoted with a superscripted 'A'. Note that for the initial timepoint the desired and achieved end-effector poses are necessarily equal: ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \homttd{R}{EE}{0} = \homtta{R}{EE}{0} = \homtt{R}{EE}{0} \] \end{document}]() as defined in Equation (S2.15).

To build a joint-space trajectory given an initial manipulator posture ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rjt{0} \] \end{document}]() and corresponding actual operational space trajectory these steps are implemented recursively:

1. Compute the Jacobian pseudo-inverse at the current manipulator posture: ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \Jpinv(\rjt{k}) \] \end{document}]().
2. Compute the next manipulator posture according to equations (S2.8) and (S2.9):

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[  \rj(t_{k+1}) = \rj(t_k) + \Jpinv(\rj(t_k)) \begin{bmatrix}  ^R\vect{p}^D(\tk{k+1}) - ^R\vect{p}^A(\tk{k})\\  \rotv[R](\rotm{R}{EE}^D(\tau) \cdot (\rotm{R}{EE}^A(\tk{k}))^T, \tk{k+1})  \end{bmatrix} \] \end{document} |  | (S2.) |

1. Compute the achieved end-effector pose using the robot's forward kinematics function:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[  \homtta{R}{EE}{k+1} = K(\rjt{k+1}) \] \end{document} |  | (S2.) |

Again, it is insightful to note that:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rotm{R}{EE}^D(\tk{k+1}) \cdot (\rotm{R}{EE}^A(\tk{k}))^T = \rotm{R}{EE}^A(\tk{k}) \cdot (\rotm{R}{EE}^A(\tk{k}))^T \cdot \rotm{R}{EE}^D(\tk{k+1}) \cdot (\rotm{R}{EE}^A(\tk{k}))^T \] \end{document} |  | (S2.) |

Equation (S2.19) demonstrates that ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rotm{R}{EE}^D(\tk{k+1}) \cdot (\rotm{R}{EE}^A(\tk{k}))^T \] \end{document}]() represents the rotation from the achieved end-effector pose at time ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \tk{k} \] \end{document}]() to the desired end-effector pose at time ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \tk{k+1} \] \end{document}]() in the robot reference frame.

1. **Optimization Problem**

Steps 1-3, along with Equations (S2.14), (S2.15), and (S2.16), provide a recursive relationship for computing a joint-space trajectory ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rjt{k} \] \end{document}]() and corresponding achieved operational space trajectory ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \homtta{R}{EE}{k} \] \end{document}]() from an initial manipulator posture ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rjt{0} \] \end{document}]() and the humerus trajectory in the motion capture reference frame ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \homtt{MC}{H}{k} \] \end{document}](). The problem at hand then is to determine an optimal initial manipulator posture ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rjt{0} \] \end{document}]() that preserves the kinetic properties of the humerus trajectory ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \homtt{MC}{H}{k} \] \end{document}]() while minimizing the effort expended by the robot. Given that the forward kinematics function, ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ K \] \end{document}](), is non-linear the problem naturally lends itself to utilizing a non-linear optimizer.

In the formulation of this study, the parameters of the optimization problem are the joint angles of the initial manipulator posture, ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rjt{0} \] \end{document}](). Since there are only six parameters, and because computation of the analytical gradient as the optimization problem is formulated is exceedingly difficult, a gradient-free non-linear numerical optimization method was selected. Specifically, the Constrained Optimization by Linear Approximation (COBYLA) method [5] as implemented in the NLopt library [6] within the Pagmo2 library [7] was utilized. The following sections describe the bounds, (in)equality constraints, and the objective function of the optimization. In all sections the degrees of freedom of the robot is ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ n \] \end{document}](), while the number of timepoints in the trajectory is ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ m \] \end{document}]().

A starting guess is necessary to seed the optimizer. This could be an educated guess based on the trajectory being optimized, or simply the home position of the robot. In our experience, the non-linear optimizer is robust. It finds the same initial manipulator posture even when the seeds are a large distance away from each other in joint space, although global optimality is not guaranteed. The provided seed (initial manipulator posture ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rjt{0} \] \end{document}]()), is evaluated for its fitness in faithfully reproducing the motion capture trajectory and minimizing the effort expended by the robot. The results are utilized as feedback in selecting the next manipulator posture for evaluation and a new optimization iteration begins. The optimization process stops when a maximum number of iterations has been reached, or when the change in ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rjt{0} \] \end{document}]() or in the objective function (described below) from one iteration to the next reaches a defined tolerance.

Note that in the following sections the rotation vector function, ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rotv \] \end{document}](), is overloaded to accept just a proper orthogonal matrix instead of needing to specify an entire trajectory and the timepoint within the trajectory. Although the prior notation elucidated the proper calculation of orientation error, practically this function only needs to accept one argument.

* 1. **Objective Function**

The objective function determines the fitness of the joint space trajectory in accomplishing an objective, namely, minimizing the robot's joint utilization (measured as a percentage of the joint velocity limits) for the entire trajectory. Specifically, the goal is to minimize Equation (S2.20) whose summand is the cost function that maps a joint velocity percentage to a cost (Fig. S2.1). Let ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rjv(t_k) = (\rj(t_{k+1}) - \rj(t_k))/\varDelta t \] \end{document}]() and ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rjv^{max} \] \end{document}]() denote the robot joint velocity limits, then:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ f_{obj}(\rjv, \rjv^{max}) = \sum_{k=0}^{m-2} \sum_{j=1}^{n} \frac{20}{1+e^{-10 \cdot (\abs{\rjvt{k}}/\rjjvm - 1.2)}} \] \end{document} |  | (S2.) |



Fig. S2.1: Plot of cost function versus joint velocity percentage

* 1. **Bounds**

The robot joint angles of initial manipulator posture are the parameters of the optimization problem. Therefore, the lower and upper bounds for the algorithm are comprised of the robot joint limits. If the lower and upper joint limits of the robot are denoted by ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rj_{l} \] \end{document}]() and ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rj_{u} \] \end{document}](), respectively, then:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rj_{l} \leq \rjt{0} \leq \rj_{u} \] \end{document} |  | (S2.) |

* 1. **Equality Constraints**
1. This equality constraint verifies that the desired and achieved operational space trajectories are congruent. It assures that there is no appreciable drift due to linearization and that solutions that pass through singularities are avoided.

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[   \begin{split}   \sum_{k=0}^{m-1} [(&\postd[R]{k} - \posta[R]{k}) \cdot (\postd[R]{k} - \posta[R]{k}) + \\ &\rotv[R](\rotm{R}{EE}^D(\tk{k}) (\rotm{R}{EE}^A(\tk{k}))^T) \cdot \rotv[R](\rotm{R}{EE}^D(\tk{k}) (\rotm{R}{EE}^A(\tk{k}))^T)] = 0   \end{split} \] \end{document} |  | (S2.) |

1. This equality constraint assures that the implied rotation from the motion capture frame to the robot frame ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \rotm{MC}{R} \] \end{document}]() happens about the gravitational axis. By only allowing rotations (and any translations) about the gravitational axis, the kinetic properties of the trajectory are conserved. Letting the z-axis coincide with the gravitational axis with ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \beta \in \mathbb{R} \] \end{document}]():

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[  \rotv[MC]{(\rotmt{MC}{EE}{0} \cdot (\rotmt{R}{EE}{0})^T)} = \begin{bmatrix}  0 \\ 0 \\ \beta  \end{bmatrix} \] \end{document} |  | (S2.) |

* 1. **Inequality Constraints**
1. Since the bounds of the optimization problem only restrict the initial manipulator posture to within the robot joint limits, inequality constraints on the joint space trajectory are utilized to respect the robot joint limits for the entire trajectory:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[  \rj_{l} \leq \rjt{k} \leq \rj_{u} \  \forall k \] \end{document} |  | (S2.) |

1. This inequality constraint assures that the joint space trajectory respects the robot joint velocity limits:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[  \abs{\rjvt{k}} \leq \rjvm: 0 \leq k \leq m-2 \] \end{document} |  | (S2.) |

1. The optimization algorithm can be instructed to stay within a rectangular prism in order to avoid obstacles in the laboratory. The boundaries of this rectangular prism are specified to the optimization algorithm via two of its corners ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \vect{c}_l \in \reals{3} \] \end{document}]() and ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \vect{c}_u \in \reals{3} \] \end{document}]() such that:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[  \vect{c}_l \leq \posta[R]{k} \leq \vect{c}_u \  \forall k \] \end{document} |  | (S2.) |

1. **Proofs**
	1. **Lemma 1**

Let ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \vect{A} \] \end{document}]() and ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \vect{R} \] \end{document}]() be proper orthogonal matrices, then:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \angv(\vect{R}(\tau) \cdot \vect{A}) = \angv(\vect{R}(\tau)) \] \end{document} |  | (S2.) |

*Proof:* For a trajectory ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \vect{R}(\tau) \] \end{document}](), the angular velocity vector ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \angv(\vect{R}(\tau)) \] \end{document}]() is the axial vector associated with the skew-symmetric angular velocity tensor ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \vect{W}_{\vect{R}}(\tau) \] \end{document}]() [8]:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \vect{W}_{\vect{R}}(\tau) = \frac{d\vect{R}(\tau)}{d\tau} \cdot \vect{R}^T(\tau) \] \end{document} |  | (S2.) |

For a trajectory ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \vect{R}(\tau) \cdot \vect{A} \] \end{document}]() the angular velocity tensor is:

|  |  |  |
| --- | --- | --- |
| %FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \begin{aligned} \vect{W}_{\vect{R} \cdot \vect{A}}(\tau) &= \frac{d(\vect{R}(\tau) \cdot \vect{A})}{d\tau} \cdot  (\vect{R}(\tau) \cdot \vect{A})^T \\ &= \frac{d(\vect{R}(\tau))}{d\tau} \cdot \vect{A} \cdot \vect{A}^T \cdot \vect{R}^T(\tau) \\ &=\vect{W}_{\vect{R}}(\tau) \end{aligned} \] \end{document} |  | (S2.) |

Since the angular velocity tensors for ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \vect{R}(\tau) \] \end{document}]() and ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \vect{R}(\tau) \cdot \vect{A} \] \end{document}]() are equal then their corresponding angular velocity axial vectors - ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \angv(\vect{R}(\tau)) \] \end{document}]() and ![%FontSize=11 %TeXFontSize=11 \documentclass{article} \pagestyle{empty} \begin{document} \[ \angv(\vect{R}(\tau) \cdot \vect{A}) \] \end{document}](), respectively, must be equal.

1. **References**

1. Siciliano B, Sciavicco L, Villani L, Oriolo G. Robotics: modelling, planning and control: Springer Science & Business Media; 2010.

2. Buss SR. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. IEEE Journal of Robotics and Automation. 2004;17(1-19):16.

3. Diebel J, editor Representing Attitude : Euler Angles , Unit Quaternions , and Rotation Vectors2006.

4. Peres A. Finite rotations and angular velocity. American Journal of Physics. 1980;48:70-1.

5. Powell MJD. A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation. In: Gomez S, Hennart J-P, editors. Advances in Optimization and Numerical Analysis. Dordrecht: Springer Netherlands; 1994. p. 51-67.

6. Johnson SG. The NLopt nonlinear-optimization package. 2018.

7. Biscani F, Izzo D, Jakob W, Märtens M, Mereta A, Kaldemeyer C, et al. esa/pagmo2: pagmo 2.10. 2019.

8. Condurache D, Matcovschi M. Computation of angular velocity and acceleration tensors by direct measurements. Acta Mechanica. 2002;153(3):147-67. doi: 10.1007/bf01177449.