Analytic Gradient of Snake Robot Inverse Kinematic Objective

For sake of clarity, we describe our derivation of the analytic gradient of a cost function for doing inverse kinematics for a simple snake robot in the form of an illustrative example of a snake with 3 links and 3 sets of join angles (parameterized as Euler angles). Let the orientation of link l_i w.r.t. the orientation of the previous link be $\boldsymbol{\theta}_i = [\alpha_i, \beta_i, \gamma_i]$. We assume that the orientation of first link is with respect to the x-axis.

Now denote the equivalent rotation matrix for link rotation θ_i as $\mathbf{R}_i$. Our objective is to identify the set of angles $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3\}$ that place the tip position of the third link $\mathbf{p}_3$ as close as possible to a target $\mathbf{t}$ with a final orientation of the third link as close to a target orientation $\mathbf{q_t}$ while avoiding a set of spherical obstacles and obeying a set of hard constraints places on the joint angles, $\{\boldsymbol{\theta}_1^{\min}, \boldsymbol{\theta}_2^{\min}, \boldsymbol{\theta}_3^{\min}\}$, $\{\boldsymbol{\theta}_1^{\max}, \boldsymbol{\theta}_2^{\max}, \boldsymbol{\theta}_3^{\max}\}$.

The overall cost function is composed of distance, orientation, and obstacle components whose proportion to the overall cost can be adjusted by a set of scaling factors λ_x, $\sum_x \lambda_x = 1$

$$C = \lambda_d C_d + \lambda_o C_o$$

We wish to compute the gradient of the overall cost with respect to each of the angle sets $\boldsymbol{\theta}_i$. Because the gradient is a linear operator, we have

$$\nabla_{\boldsymbol{\theta}_i} C = \nabla_{\boldsymbol{\theta}_i} \lambda_d C_d + \nabla_{\boldsymbol{\theta}_i} \lambda_o C_o$$

The distance cost is the sum of squared differences of the final effector position $\boldsymbol{p_3}$ and target $\boldsymbol{t}$, where $\boldsymbol{p_j}$ denotes the final effector position after each successive rotation j.

$$C_d = (\mathbf{p}_3 - \boldsymbol{t})^{\mathbf{T}} (\mathbf{p}_3 - \mathbf{t})$$

And its gradient with respect to a set of joint angles,

$$\nabla_{\boldsymbol{\theta}_i} C_d = \nabla_{\mathbf{p}_3} C_d J_{\mathbf{p}_3}(\boldsymbol{\theta}_i)$$

First note that $\mathbf{p}_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^{\mathbf{T}}$ and $\mathbf{l}_i = \begin{bmatrix} l_i & 0 & 0 \end{bmatrix}^{\mathbf{T}}$. We can make use of the recursive relationship of end effector position between rotations.

$$\begin{aligned}
\mathbf{p}_0 &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^{\mathbf{T}} \\
\mathbf{p}_1 &= \mathbf{R}_3(\mathbf{p}_0 + \mathbf{l}_3) \\
\mathbf{p}_2 &= \mathbf{R}_2(\mathbf{p}_1 + \mathbf{l}_2) \\
\mathbf{p}_3 &= \mathbf{R}_1(\mathbf{p}_2 + \mathbf{l}_1)
\end{aligned}$$

Therefore we can simply compute the Jacobian $J_{\mathbf{p}_3}(\boldsymbol{\theta}_i)$ using the chain rule

$$\begin{aligned}
J_{\mathbf{p}_3}(\theta_1) &= J_{\mathbf{p}_3}(\theta_1) \\
J_{\mathbf{p}_3}(\theta_2) &= J_{\mathbf{p}_3}(\mathbf{p}_2) J_{\mathbf{p}_2}(\theta_2) \\
J_{\mathbf{p}_3}(\theta_3) &= J_{\mathbf{p}_3}(\mathbf{p}_2) J_{\mathbf{p}_2}(\mathbf{p}_1) J_{\mathbf{p}_1}(\theta_3)
\end{aligned}$$

More generally this is,

$$J_{\mathbf{p}_n}(\theta_i) = \left[\prod_{k=n-i}^{n} J_{\mathbf{p}_k}(\mathbf{p}_{k-1}) \right] J_{\mathbf{p}_{n-i+1}}(\theta_i)$$

Where

$$J_{\mathbf{p}_k}(\mathbf{p}_{k-1}) = \mathbf{R}_{n-k+1}$$

We also find (Using symbolic MATLAB) that

$$J_{\mathbf{p}}(\theta)[1,1] = \sin(\alpha)\sin(\beta) + \cos(\alpha)\cos(\gamma)\sin(\beta))(l_2 + p_2) + (\cos(\alpha)\sin(\gamma) - \cos(\gamma)\sin(\alpha)\sin(\beta))(l_3 + p_3)$$
$$J_{\mathbf{p}}(\theta)[1,2] = \cos(\alpha)\cos(\beta)\cos(\gamma)(l_3 + p_3) - \cos(\gamma)\sin(\beta)(l_1 + p1) + \cos(\beta)\cos(\gamma)\sin(\alpha)(l_2 + p_2)$$
$$J_{\mathbf{p}}(\theta)[1,3] = (\cos(\gamma)\sin(\alpha) - \cos(\alpha)\sin(\beta)\sin(\gamma))(l_3 + p_3) - (\cos(\alpha)\cos(\gamma) + \sin(\alpha)\sin(\beta)\sin(\gamma))(l_2 + p_2) - \cos(\beta)\sin(\gamma)(l_1 + p1)$$
$$J_{\mathbf{p}}(\theta)[2,1] = -(\cos(\gamma)\sin(\alpha) - \cos(\alpha)\sin(\beta)\sin(\gamma))(l_2 + p_2) - (\cos(\alpha)\cos(\gamma) + \sin(\alpha)\sin(\beta)\sin(\gamma))(l_3 + p_3)$$
$$J_{\mathbf{p}}(\theta)[2,2] = \cos(\beta)\sin(\alpha)\sin(\gamma)(l_2 + p_2) - \sin(\beta)\sin(\gamma)(l_1 + p1) + \cos(\alpha)\cos(\beta)\sin(\gamma)(l_3 + p_3)$$
$$J_{\mathbf{p}}(\theta)[2,3] = (\sin(\alpha)\sin(\gamma) + \cos(\alpha)\cos(\gamma)\sin(\beta))(l_3 + p_3) - (\cos(\alpha)\sin(\gamma) - \cos(\gamma)\sin(\alpha)\sin(\beta))(l_2 + p_2) + \cos(\beta)\cos(\gamma)(l_1 + p1)$$
$$J_{\mathbf{p}}(\theta)[3,1] = \cos(\alpha)\cos(\beta)(l_2 + p_2) - \cos(\beta)\sin(\alpha)(l_3 + p_3)$$
$$J_{\mathbf{p}}(\theta)[3,2] = -\cos(\beta)(l_1 + p1) - \cos(\alpha)\sin(\beta)(l_3 + p_3) - \sin(\alpha)\sin(\beta)(l_2 + p_2)$$
$$J_{\mathbf{p}}(\theta)[3,3] = 0$$

The obstacle cost can be derived in a similar fashion using the chain rule. We are trying to minimize the distance from the midpoint of each link to the point closest to the surface of each spherical obstacle.