Pose Generation Algorithm

Akash Patel
Krang Lab Group

***Abstract*— //TODO**

I. PURPOSE

The goal of this algorithm is to generate a good set of training poses. A ”good” set of poses is defined with the following metric. Performing a learning algorithm (gradient descent in this case) on this set of poses should result in convergence of all reasonable initial βs and the converged values should have good predictions for a test set of poses.

II. EXPLANATION

A. Random Pose Generation

The first step is to generate a set of poses to filter. This is done through random sampling of the entire joint space, specifically the subset of the joint space in which the pose is balanced and safe. **Algorithm 1** describes the approach implemented.

1) Balanced: A balanced pose is defined to be a pose in which the x_{COM} is close to zero. The pose's x_{COM} is determined using DART. Poses in which their x_{COM} is lower than some threshold are then considered balanced. The threshold used in our case is 1 mm. The balanced pose of any pose is determined using `nlopt`, an optimization library. //TODO can explain the optimization in more detail

2) Safe: A safe pose is defined to be a pose that has no collision. This includes self-collision as well as collision with a flat ground. Collision is implemented using DART's collision detector with a 3D robot model as input. //TODO can explain collision boxes of the model

B. Filtering

After we obtain a distribution of randomly sampled balanced and safe poses we need to determine which poses from this set are ”good” poses. A filtering algorithm is devised based on discarding poses with a relatively small gradient and learning on the poses which result in a large gradient.

At every update step, the pose with the largest total difference of the predicted x_{COM} value ($\Phi_i\beta$) is used. This ensures that only poses with a large enough stepsize are used and not poses in which the βs values would change very little. The stopping condition for this filtering is either if we run out of poses to learn on or that any learning we do results in an x_{COM} of less than the x_{COM} tolerance level decided (such as 2 mm).

Algorithm 2 describes the filtering procedure.

III. COMPLETE ALGORITHM

Algorithm 1 Pose Generation

Input: n_{poses}
Output: $\bar{q}$ (balanced and safe set of poses)

1: $j = 0$
2: **while** $j < n_{poses}$ **do**
3: q_j = Randomly generated pose
4: $\bar{q}_i$ = balanced pose of q_j
5: **if** $\bar{q}_i$'s $x_{com} <= x_{tol}$ and $\bar{q}_i$ is safe **then**
6: Add $\bar{q}_i$ to $\bar{q}$
7: **end if**
8: $j = j + 1$
9: **end while**
10: **return** $\bar{q}$

Algorithm 2 Pose Filtering

Input: $\bar{q} \in \mathbb{R}^{n_{DOF} \times n_{poses}}$ (input set of poses),
$\Phi \in \mathbb{R}^{n_{poses} \times dim(\beta)}$ ($\phi(q)$ evaluated at each given pose),
$\bar{\beta} \in \mathbb{R}^{dim(\beta) \times n_\beta}$ (set of initial β)
Output: $\tilde{q}$ (filtered set of poses)

1: $j = 0$
2: **while** $j < n_{poses}$ **do**
3: $i^* = \text{argmax}_i \sum_k |\Phi_i \beta_k|$
4: $\phi^* = \Phi_{i^*}$
5: $j = j + 1$
6: $\tilde{q}_j = \bar{q}_{i^*}$
7: $\beta_k = \beta_k - \eta \cdot \phi^* \beta_k \cdot \phi^* \quad \forall \quad k \in \{1, ..., n_\beta\}$
8: **if** $|\phi^* \beta_k| < x_{tol} \quad \forall \quad k \in \{1, ..., n_\beta\}$ for last few iterations **then**
9: go to step 15
10: **else**
11: $\Phi = \Phi$ without Φ_{i^*}
12: $\bar{q} = \bar{q}$ without $\bar{q}_{i^*}$
13: **end if**
14: **end while**
15: **return** $\bar{q}$

IV. EXPERIMENT

//TODO

V. RESULTS

//TODO Add some good plots