Algorithm for predicting the pose of a gecko-inspired soft robot for a given reference input

Lars Schiller

July 9, 2018

Due to the fact that the behaviour of soft materials is difficult to predict with conventional methods, an algorithm based on a geometric optimization problem is presented. The algorithm can be used to predict the actual pose of the robot for a given reference input. Figure 1 is taken as an example. The initial position is shown in black. The individual limbs of the robot have a certain bending angle and all feet are fixed. Now the torso of the robot should be actuated. If only the bending angle of the torso is changed, the grey dashed pose is obtained. Obviously, the two rear feet are no longer in the same position. Since these feet are fixed, the robot will behave differently in reality. In fact, it's much more likely to take up the grey pose. Although the bending angles of all limbs have changed, the condition that all feet remain motionless has been fulfilled.

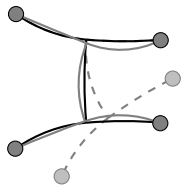

Figure 1: Example Usage

1 Predicting the pose for a given reference input

In order to let the robot take a pose the user has nine degrees of freedom at his disposal: the correspondending pressures for the five bending angles α_i of the limbs $i = 1, \ldots, 5$ and the state of the fixation actuators $f_j \in \{0, 1\}$ of the feet $j = 1, \ldots, 4$. For the unloaded state, a calibration function can be formulated for each limb, which maps the input pressure on the bending angle (under load, this function no longer needs to be valid). But the input pressure can be seen as an reference for the bending angle. Accordingly, a reference input $\boldsymbol{r}$ can be described by

$$\boldsymbol{\alpha}_{\text{ref}} = [\ \alpha_{\text{ref},1}\ \alpha_{\text{ref},2}\ \alpha_{\text{ref},3}\ \alpha_{\text{ref},4}\ \alpha_{\text{ref},5}\ ]^\top \tag{1}$$

$$\boldsymbol{f} = [\ f_1\ f_2\ f_3\ f_4\ ]^\top \tag{2}$$

$$\boldsymbol{r} = \left[\ \boldsymbol{\alpha}_{\text{ref}}^\top\ \boldsymbol{f}^\top\ \right]^\top. \tag{3}$$

However, this information is not sufficient to describe the robot's actual pose. Experiments have shown that the bending angle of a limb can vary significantly at the same pressure level due to the softness of the used material. The length of a limb can also differ. In order to describe the pose of the robot, the actual bending angles $\boldsymbol{\alpha}$:

$$\boldsymbol{\alpha} = [\alpha_1\ \alpha_2\ \alpha_3\ \alpha_4\ \alpha_5]^\top, \tag{4}$$

the actual lengths of the individual limbs $\boldsymbol{\ell}$:

$$\boldsymbol{\ell} = [\ell_1\ \ell_2\ \ell_3\ \ell_4\ \ell_5]^\top, \tag{5}$$

and the orientation of the robot's center point ε must be known (see Fig. 2). These quantities are defined as the variable to be optimized:

$$\boldsymbol{x} = \left[\ \boldsymbol{\alpha}^\top\ \boldsymbol{\ell}^\top\ \varepsilon\ \right]^\top. \tag{6}$$

Furthermore, the position of (at least) one fixed foot ($f_j \stackrel{!}{=} 1$) must be known. Then, the pose of the robot $\boldsymbol{\rho}$ can be determined under the assumption of a constant curvature by drawing arcs with the corresponding lengths and

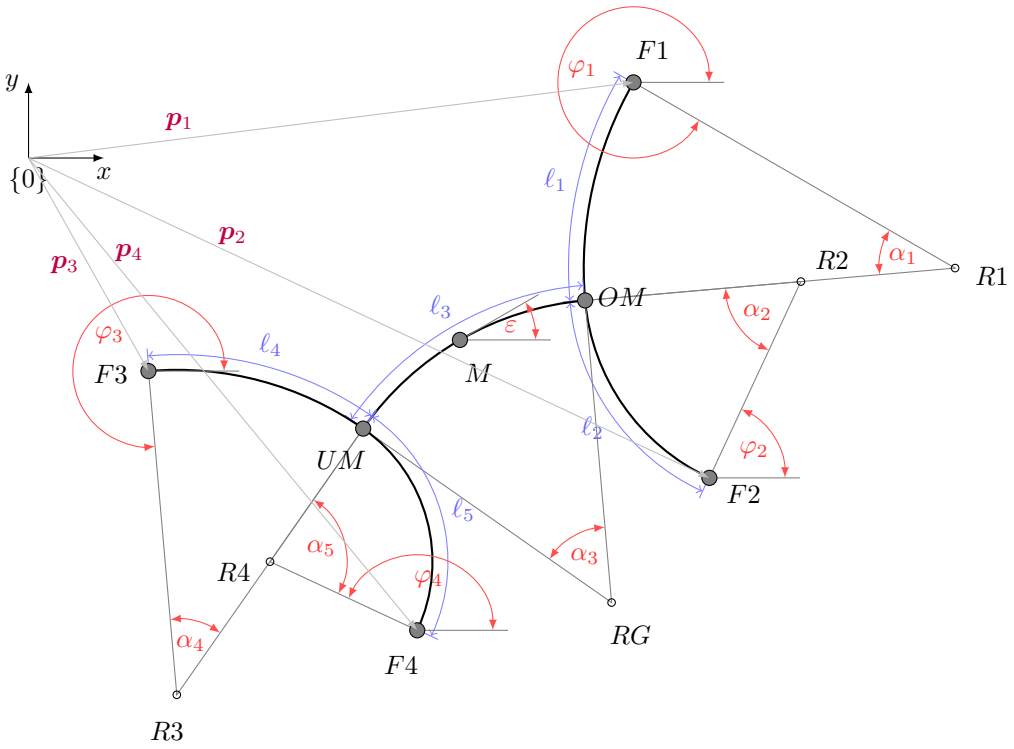

Figure 2: Nomencalture

angles. A pose can therefore be described in generally as a function of $\boldsymbol{x}$ and the position $\boldsymbol{P} = [\boldsymbol{p}_1\ \boldsymbol{p}_2\ \boldsymbol{p}_3\ \boldsymbol{p}_4]^\top \in \mathbb{R}^{4\times 2}$ and state $\boldsymbol{f}$ of all feet:

$$\boldsymbol{\rho} = [\boldsymbol{x},\ \boldsymbol{P},\ \boldsymbol{f}]\,. \tag{7}$$

For a given feasible initial pose, the next pose must be determined so that all fixed feet do not move. This can be achieved within a certain margin by deviating the bending angle from the reference angle and deviating the actual length from the nominal length. To describe this mathematically, a new index k is introduced, which assigns the quantities to a specific time step. The new positions of the fixed feet $\boldsymbol{P}_k$ are assumed to be the positions from the previous pose $\boldsymbol{P}_{k-1}$. This can be used to define the constraint for the next pose. All newly fixed feet must have the same position as in the previous step:

$$\left|\left|\mathrm{diag}(\boldsymbol{f}_k)(\boldsymbol{P}_k - \boldsymbol{P}_{k-1})\right|\right|_2 \stackrel{!}{=} 0, \tag{8}$$

It has already been mentioned that the bending angles $\boldsymbol{\alpha}$ and the lengths of the limbs $\boldsymbol{\ell}$ are quite variable. By defining

$$\boldsymbol{\ell}_n = [\ell_{n,1}\ \ell_{n,2}\ \ell_{n,3}\ \ell_{n,4}\ \ell_{n,5}]^\top \tag{9}$$

as the vector containing the nominal length of each limb, it is possible to quantify the length deviation. The orientation angles of the fixation actuators $\boldsymbol{\varphi}$ also have a certain margin. The value of the orientation angles can be calculated as a function of $\boldsymbol{\alpha}$ and ε (well, basically as a function of $\boldsymbol{x}$):

$$\boldsymbol{\varphi}(\boldsymbol{\alpha}, \varepsilon) = (16), \tag{10}$$

where the exact formula is given in the appendix 16. Now a objective function can be formulated which quantifies the deviations of length, angle and orientation:

$$obj(\boldsymbol{x}_k) = w_{\text{len}}\left|\boldsymbol{\ell}_k - \boldsymbol{\ell}_n\right|_2 + w_{\text{ang}}\left|\boldsymbol{\alpha}_k - \boldsymbol{\alpha}_{\text{ref},k}\right|_2 + w_{\text{ori}}\left|\mathrm{diag}(\boldsymbol{f}_k)(\boldsymbol{\varphi}_k - \boldsymbol{\varphi}_{k-1})\right|_2. \tag{11}$$

The weighting factors can be interpreted physically. To do so, the weighting factor w_{len} describes the elasticity of the limbs and w_{ang} the bending stiffness of the limbs. The term weighted by w_{ori} describes the difference between the orientation of the newly fixed feet compared to the orientation in the previous time step. This can be seen as a dimension for the torsional stiffness of the fixation actuators. The new pose can now be determined by solving the non-linear optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{x}_k \in \mathcal{X}}\ obj(\boldsymbol{x}_k) \\ &\textit{subjected to}\quad \left|\left|\mathrm{diag}(\boldsymbol{f}_k)(\boldsymbol{P}_k - \boldsymbol{P}_{k-1})\right|\right|_2 = 0. \end{aligned} \tag{12}$$

Here $\mathcal{X}$ describes the set of allowed values. Each quantity inside $\boldsymbol{x}$ has bounds, which are given in the following table:

var	$\boldsymbol{\alpha}$	$\boldsymbol{\ell}$	ε
bounds	$[\boldsymbol{\alpha}_{\text{ref}} - b_{\text{ang}},\ \boldsymbol{\alpha}_{\text{ref}} + b_{\text{ang}}]$	$[(1-b_{\text{len}})\boldsymbol{\ell}_n,\ (1+b_{\text{len}})\boldsymbol{\ell}_n]$	$[0°,\ 360°]$

These bounds can be tuned with the scalars b_{ang} and b_{len}. For solving the problem, for example the `SLSQP`-Algorithm provided by the python package `scipy.optimize` can be used. Note that the evaluation of the objective function (11) is quite cheap. The expensive part is the evaluation of the constraint function (8), since the calculation of all feet positions for a given $\boldsymbol{x}$ is opulent and outlined in the appendix (17).

In summary, it is possible to set up a function that can predict the next pose of the robot, depending on the reference input and the previous pose:

$$\boldsymbol{\rho}_k = \text{fun}\left(\boldsymbol{r}, \boldsymbol{\rho}_{k-1}\right) \tag{13}$$

This function can be tuned by the parameters given in the following table:

param	description
b_{ang}	allowed absolute deviation of the bending angle to the reference angle
b_{len}	allowed percentage deviation of the length to the nominal length
w_{len}	costs of the length deviation
w_{ang}	costs of the bending angle deviation
w_{ori}	costs of the orientation angle deviation

2 Finding optimal gait patterns

The presented algorithm can now be used to find new gait patterns. Since it provides the ability to predict the robot's pose for a given reference input, it is able to predict all associated poses to a sequence of reference inputs by recursive application.

A gait pattern consists of a sequence of poses taken in a continuous loop. To find an optimal sequence of reference inputs, it is first necessary to define what is optimal. This depends entirely on the type of running pattern you are looking for. Two examples are presented below.

2.1 Straight Gait

Soll der Roboter geradeaus Laufen, so ist es optimal die Distanz von Startpunkt und Endpunkt möglichst zu maximieren.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r}_1^\top \\ \boldsymbol{r}_2^\top \end{bmatrix} = \begin{bmatrix} \alpha_{\text{ref},1} & \alpha_{\text{ref},2} & \alpha_{\text{ref},3} & \alpha_{\text{ref},4} & \alpha_{\text{ref},5} & 1 & 0 & 0 & 1 \\ \alpha_{\text{ref},2} & \alpha_{\text{ref},1} & -\alpha_{\text{ref},3} & \alpha_{\text{ref},5} & \alpha_{\text{ref},4} & 0 & 1 & 1 & 0 \end{bmatrix} \tag{14}$$

2.2 Gait Pattern for a curve

A Appendix

Coordinates for $F1$ fixed:

$$r_i = \frac{360\ \ell_i}{2\pi\ \alpha_i}, \quad \text{for } i \in [1,2,3,4,5] \tag{15}$$

$$\boldsymbol{\varphi} = \begin{bmatrix} \varepsilon - \alpha_1 - \frac{1}{2}\alpha_3 \\ \varphi_1 + \alpha_1 + \alpha_2 \\ 180° + \alpha_3 - \alpha_2 + \alpha_4 + \varphi_2 \\ 180° + \alpha_3 - \alpha_1 + \alpha_5 + \varphi_1 \end{bmatrix} \tag{16}$$

$$R1 = \begin{bmatrix} F1_x + \cos(c_1)\ r_1 \\ F1_y + \sin(c_1)\ r_1 \end{bmatrix} \tag{17}$$

$$OM = \begin{bmatrix} R1_x - \sin(90 - c_1 - \alpha_1)\ r_1 \\ R1_y - \cos(90 - c_1 - \alpha_1)\ r_1 \end{bmatrix} \tag{18}$$

$$RG = \begin{bmatrix} OM_x + \cos(90 - c_1 - \alpha_1)\ r_\gamma \\ OM_y - \sin(90 - c_1 - \alpha_1)\ r_\gamma \end{bmatrix} \tag{19}$$

$$UM = \begin{bmatrix} RG_x - \cos(\gamma - 90 + c_1 + \alpha_1)\ r_\gamma \\ RG_y - \sin(\gamma - 90 + c_1 + \alpha_1)\ r_\gamma \end{bmatrix} \tag{20}$$

$$R4 = \begin{bmatrix} UM_x + \sin(\gamma - 90 + c_1 + \alpha_1)\ r_4 \\ UM_y - \cos(\gamma - 90 + c_1 + \alpha_1)\ r_4 \end{bmatrix} \tag{21}$$

$$F4 = \begin{bmatrix} R4_x + \sin(-c_4 - 90)\ r_4 \\ R4_y + \cos(-c_4 - 90)\ r_4 \end{bmatrix} \tag{22}$$

$$R3 = \begin{bmatrix} UM_x + \sin(\gamma - 90 + c_1 + \alpha_1)\ r_3 \\ UM_y - \cos(\gamma - 90 + c_1 + \alpha_1)\ r_3 \end{bmatrix} \tag{23}$$

$$F3 = \begin{bmatrix} R3_x - \cos(-c_3)\ r_3 \\ R3_y + \sin(-c_3)\ r_3 \end{bmatrix} \tag{24}$$

$$R2 = \begin{bmatrix} OM_x + \cos(c_1 + \alpha_1)\ r_2 \\ OM_y + \sin(c_1 + \alpha_1)\ r_2 \end{bmatrix} \tag{25}$$

$$F2 = \begin{bmatrix} R2_x + \sin(c_2 - 90)\ r_2 \\ R2_y - \cos(c_2 - 90)\ r_2 \end{bmatrix} \tag{26}$$

Coordinates for $F2$ fixed:

$$R2 = \begin{bmatrix} F2_x - \sin(c_2 - 90)\ r_2 \\ F2_y + \cos(c_2 - 90)\ r_2 \end{bmatrix} \tag{27}$$

$$OM = \begin{bmatrix} R2_x - \sin(90 - c_2 + \beta_1)\ r_2 \\ R2_y - \cos(90 - c_2 + \beta_1)\ r_2 \end{bmatrix} \tag{28}$$

$$RG = \begin{bmatrix} OM_x + \cos(90 - c_2 + \beta_1)\ r_\gamma \\ OM_y - \sin(90 - c_2 + \beta_1)\ r_\gamma \end{bmatrix} \tag{29}$$

$$UM = \begin{bmatrix} RG_x - \cos(\gamma - 90 + c_2 - \beta_1)\ r_\gamma \\ RG_y - \sin(\gamma - 90 + c_2 - \beta_1)\ r_\gamma \end{bmatrix} \tag{30}$$

$$R4 = \begin{bmatrix} UM_x + \sin(\gamma - 90 + c_2 - \beta_1)\ r_4 \\ UM_y - \cos(\gamma - 90 + c_2 - \beta_1)\ r_4 \end{bmatrix} \tag{31}$$

$$F4 = \begin{bmatrix} R4_x + \sin(-c_4 - 90)\ r_4 \\ R4_y + \cos(-c_4 - 90)\ r_4 \end{bmatrix} \tag{32}$$

$$R3 = \begin{bmatrix} UM_x + \sin(\gamma - 90 + c_2 - \beta_1)\ r_3 \\ UM_y - \cos(\gamma - 90 + c_2 - \beta_1)\ r_3 \end{bmatrix} \tag{33}$$

$$F3 = \begin{bmatrix} R3_x - \cos(-c_3)\ r_3 \\ R3_y + \sin(-c_3)\ r_3 \end{bmatrix} \tag{34}$$

$$R1 = \begin{bmatrix} OM_x + \sin(90 - c_2 + \beta_1)\ r_1 \\ OM_y + \cos(90 - c_2 + \beta_1)\ r_1 \end{bmatrix} \tag{35}$$

$$F1 = \begin{bmatrix} R1_x - \sin(90 - c_1)\ r_1 \\ R1_y - \cos(90 - c_1)\ r_1 \end{bmatrix} \tag{36}$$