Goal-Oriented Graphics Animation:

Animating an Articulated Object

1 Overview

In the previous assignment, you implemented the forward-kinematics function of the robot arm, and used it to draw robot arm at at a few poses given a set of joint angles of your choice. The robot arm is shown in Figure 1.

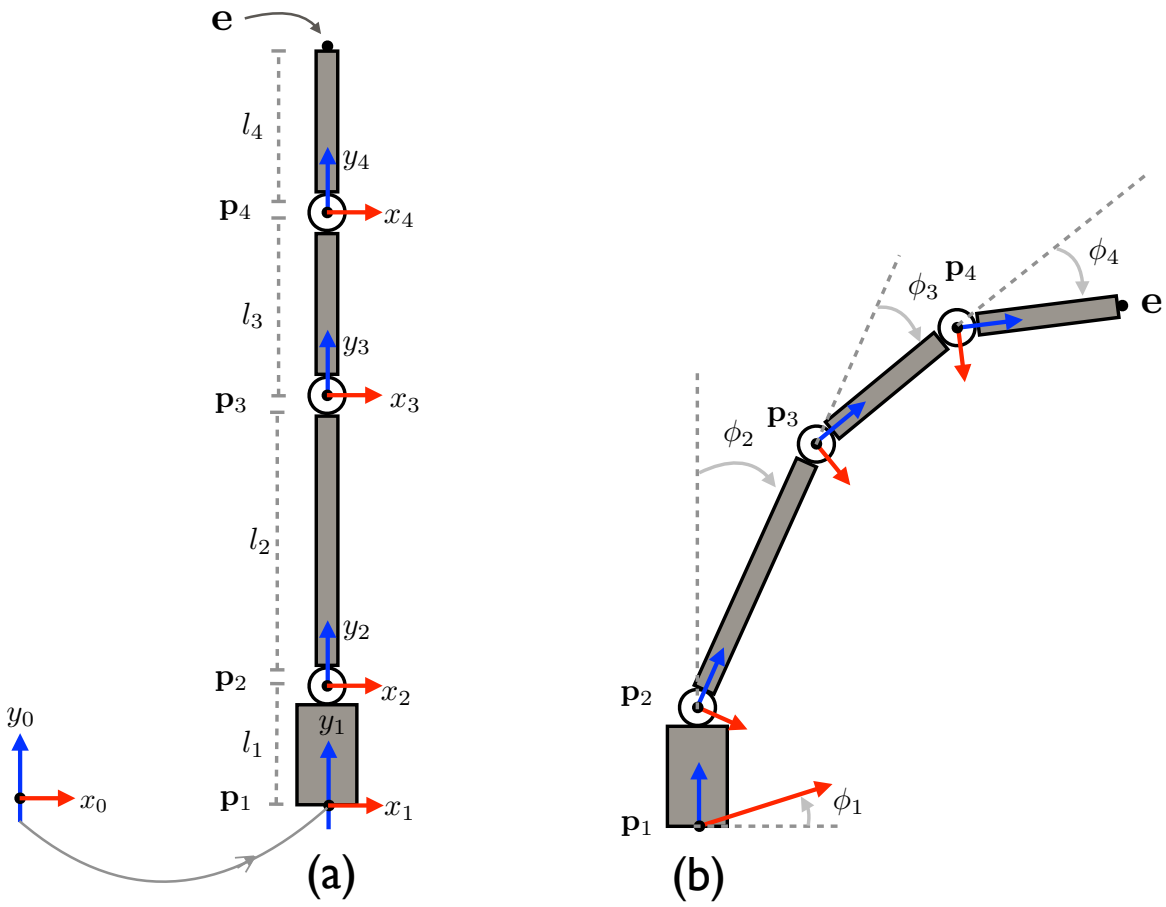


Figure 1: A 4-part robot arm. (a) Robot arm in its home (i.e., zero) configuration, i.e., all joint angles are zero. The arm part that connects to the joint is aligned with that part's y-axis. (b) Part l_1 is fixed, i.e., $\phi_1 = 0$ and the part does not rotate. All other parts have joints that rotate. The joint center $\mathbf{p}_5$ is the end-effector position $\mathbf{e}$.

In this assignment, we will implement the animation of the arm using inverse kinematics and cost-function minimization. During the animation, the arm will move from an initial position to a set of multiple goal positions. As the arm moves, it should avoid collisions with obstacles that will be placed between the initial location of the end effector and the goal position. The animation method to be implemented is described in Example 2 of Breen's paper [1].

These notes describe the main concepts and mathematical background needed to complete the assignment. Further reading might be needed to fill in some gaps. These notes were heavily based on the following coursework and tutorial materials: Steve Rotenberg's lectures slides from the CSE 169 Computer Animation course[1], Samuel Buss' notes on Inverse Kinematics [2], and from the online tutorial by Alan Zucconi [3].

2 Representing the robot arm

The robot arm is represented by a set of connected parts. Each part has its own coordinate frame. The coordinate frames are represented by transformation matrices written with respect to that frame's immediately antecedent frame of reference, starting from the global frame of reference. To build the arm's mathematical representation, our first step is to build the individual change-of-coordinate transformations (i.e., rigid-body transformation) that convert points from frame i to frame $i-1$. Following the relationship between frames of reference in a kinematic chain, the frame of reference of a given part is represented with respect to its previous part by a transformation matrix consisting of a rotation followed by a translation. Both the rotation and translation are with respect to the previous coordinate frame.

Here, we will denote the global frame of reference by Frame $\{0\}$, which is the x_0y_0 coordinate system in Figure 1. From the figure, we see that the first part of the robot arm is translated away from the global frame of frame of reference by a vector $\mathbf{p}_1 = (t_x, t_y)^\mathsf{T}$. Part 1 (and its local frame) does not rotate with respect to Frame $\{0\}$ (i.e., $\phi_1 = 0$). Thus, Frame $\{1\}$ is represented w.r.t. Frame $\{0\}$ by the following transformation:

$$T_{0,1} = \begin{bmatrix} \mathbb{1}_{2\times 2} & \mathbf{p}_1 \\ \mathbf{0}_{1\times 2} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where $\mathbb{1}$ is the identity matrix (i.e., zero-degree rotation).

[1] https://cseweb.ucsd.edu/classes/sp16/cse169-a/sessions.html

[2] https://groups.csail.mit.edu/drl/journal_club/papers/033005/buss-2004.pdf

[3] https://www.alanzucconi.com/2017/04/10/robotic-arms/

The next three parts of the robot arm and their frames, i.e., Frame {2} to Frame {4} are rotated and translated with respect to their immediately antecedent parts. The translation is by an amount equal to the length of the previous part, i.e., the matrix representing Frame {2} w.r.t. to Frame {1} is given by:

$$T_{1,2} = \begin{bmatrix} R(\phi_2) & \mathbf{p}_2 \\ \mathbf{0}_{1\times 2} & 1 \end{bmatrix} = \begin{bmatrix} \cos\phi_2 & -\sin\phi_2 & 0 \\ \sin\phi_2 & \cos\phi_2 & l_1 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2}$$

where l_1 is the length of Part 1. Similarly, Frame {3} (w.r.t. to Frame {2}) is given by:

$$T_{2,3} = \begin{bmatrix} R(\phi_3) & \mathbf{p}_3 \\ \mathbf{0}_{1\times 2} & 1 \end{bmatrix} = \begin{bmatrix} \cos\phi_3 & -\sin\phi_3 & 0 \\ \sin\phi_3 & \cos\phi_3 & l_2 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3}$$

and Frame {4} (w.r.t. to Frame {3}) is:

$$T_{3,4} = \begin{bmatrix} R(\phi_4) & \mathbf{p}_4 \\ \mathbf{0}_{1\times 2} & 1 \end{bmatrix} = \begin{bmatrix} \cos\phi_4 & -\sin\phi_4 & 0 \\ \sin\phi_4 & \cos\phi_4 & l_3 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

Finally, we need to define the frame of the end effector, i.e., Frame {5}. While we could choose to rotate the end-effector (e.g., wrist rotation), we will simply the arm configuration so that the end effector will be fixed (i.e., zero rotation). As a result, we can write Frame {5} w.r.t. its previous frame as follows:

$$T_{4,5} = \begin{bmatrix} \mathbb{1}_{2\times 2} & \mathbf{p}_5 \\ \mathbf{0}_{1\times 2} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & l_4 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5}$$

In these notes, we will use $\mathbf{p}_5$ to represent the end effector in local coordinates and use $\mathbf{e}$ to represent the end effector in global coordinates.

3 The arm's forward-kinematic function

The spatial configuration (i.e., position) of the robot arm in terms of its joint angles is given by the forward-kinematics function, which is given by:

$$\mathbf{e} = f(\mathbf{\Phi}). \tag{6}$$

This function takes the vector of joint angles $\mathbf{\Phi}$ as input and returns a vector with the location of the end-effector $\mathbf{e}$ in global coordinates. Here, vector $\mathbf{\Phi} = (\phi_1, \phi_2, \ldots, \phi_M)^\mathsf{T}$ contains the M joint angles of the kinematic object and $\mathbf{e} = (e_1, e_2, \ldots, e_N)^\mathsf{T}$ represent the N degrees-of-freedom (DOF) of the end effector. In this assignment, we will only use the position of the end effector in 2-D. As a result, $N = 2$ and we will write that position as $\mathbf{e} = (e_x, e_y)^\mathsf{T}$, again given in global coordinates. Also, our 2-D robot arm (Figure 1) has 4 joint angles, i.e., $\mathbf{\Phi} = (\phi_1, \phi_2, \phi_3, \phi_4)^\mathsf{T}$. To implement function in Equation 6, we calculate the matrix of last frame in the kinematic chain, i.e.:

$$T_{0,5} = T_{0,1}T_{1,2}T_{2,3}T_{3,4}T_{4,5} = \begin{bmatrix} R_{0,5} & \mathbf{t}_{0,5} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{7}$$

Note that, once we calculate the transformation $T_{0,5}$, its translation component, $\mathbf{t}_{0,5}$, is the location of the end-effector written in global coordinates. Here, we denote the end-effector position by $\mathbf{e} = (e_x, e_y)^\mathsf{T}$, and re-write $T_{0,5}$ as:

$$T_{0,5} = \begin{bmatrix} R_{0,5} & \mathbf{e} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{8}$$

Algorithm 1 summarizes the implementation of the forward-kinematic function.

Algorithm 1 Forward-Kinematics Function (end-effector)

1: **function** $\mathbf{e}$=f($\mathbf{\Phi}$)

Require: $l_1, \ldots, l_4$ ▷ Lengths of each part of the arm

Require: $T_{i-1,i}$, for $i = 1, \ldots, 5$. ▷ Local frame transformations

2:

3: $T_{0,5} \leftarrow T_{0,1}T_{1,2}T_{2,3}T_{3,4}T_{4,5}$.

return e ▷ Translation component of $T_{0,5}$

4: **end function**

4 Animation by cost minimization

The animation is achieved by minimizing the cost of moving the robot arm towards a goal location while encountering in its way a number of obstacles, and also having some limits placed on the range of motion of its joints. We begin by defining the cost of moving the robot arm as follows:

$$C(\mathbf{\Phi}) = \underbrace{\|\mathbf{e} - \mathbf{g}\|}_{\text{goal attraction}} + \underbrace{\sum_{i=1}^{n} \mathcal{F}_R(\|\mathbf{e} - \mathbf{o}_i\|)}_{\text{obstacle-avoidance penalty}} + \underbrace{\sum_{j=1}^{4} \mathcal{L}(\phi_j)}_{\text{Joint-range limit}}. \tag{9}$$

Here, $\mathbf{g}$ is the *goal location*, $\mathbf{o}_i$ is the location of obstacle i. Vector $\mathbf{e} = (e_x, e_y)^\mathsf{T}$ is the end-effector location in global coordinates, which is computed using the arm's *forward kinematics* function, i.e., $\mathbf{e} = f(\mathbf{\Phi})$, from Equations 7 and 8. Function $\mathcal{F}_R$ is a collision-avoidance penalty field. It penalizes poses that take the end effector too close to an obstacle, i.e., beyond a pre-defined distance R. The third component of Equation 9 limits the range of each joint angle. Function $\mathcal{L}$ is another penalty function. Its value increases as the joint angle ϕ_j approaches its maximum or minimum limit. Outside these limits, $\mathcal{L}$ vanishes.

To animate the arm, we will use the gradient-descent algorithm to solve t following minimization problem:

$$\hat{\mathbf{\Phi}} = \arg\min_{\Phi} C(\mathbf{\Phi}), \tag{10}$$

Note that all three terms of Equation 9 are actually functions of the set of joint angles Φ. In the first two terms, the dependency on Φ is implicit because $\mathbf{e}$ is given as a function of Φ (i.e., forward kinematics).

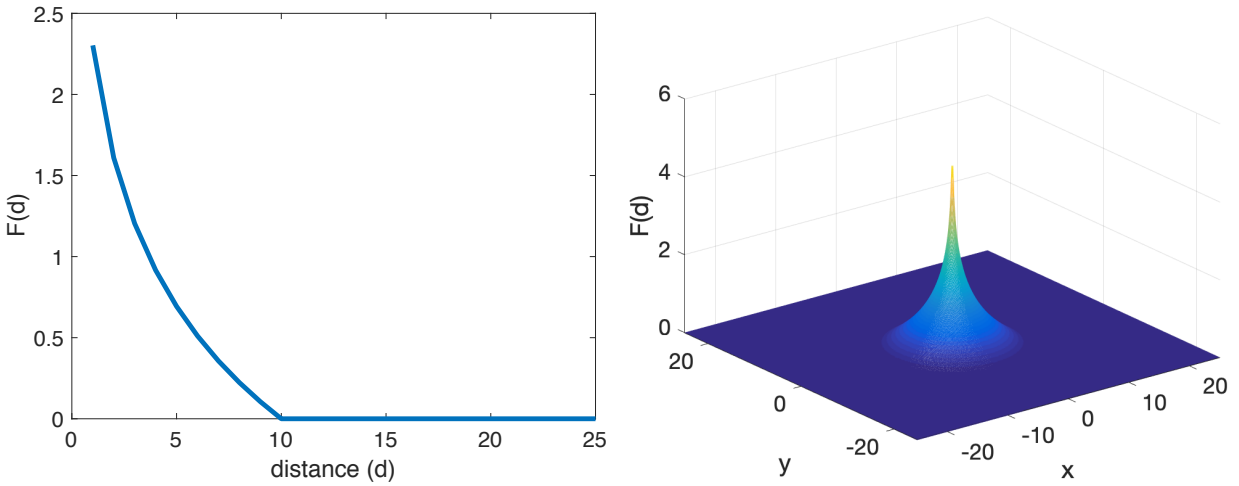
Next, we describe the two last components of Equation 9 in more detail.

4.1 Field potential function for obstacle avoidance

The field potential is a *penalty term* that increases its value as the particle approaches an obstacle. It can be defined as follows [1]:

$$\mathcal{F}_R(d) = \begin{cases} \ln(R/d), & 0 < d \leq R, \\ 0, & d > R. \end{cases} \tag{11}$$

An example of the field potential in (11) is shown in Figure 2.



(a) (b)

Figure 2: Penalty field function for $R = 10$. (a) One-dimensional plot of the penalty function. Penalty value increases for distances close to the obstacle. (b) 2-D representation of the field function for $d = \sqrt{x^2 + y^2}$, i.e., distance from any point (x, y) to the origin. When the distance from obstacle is larger than R, the field potential vanishes.

4.2 Limit function

The limit function constrains the range of motion of the joints (i.e., angles), and is given by:

$$\mathcal{L}(\phi) = \begin{cases} \ln\left(\delta / \left(\phi - \phi_{\text{min}}\right)\right), & \phi_{\text{min}} < \phi \leq \phi_{\text{min}} + \delta \\ 0, & \phi_{\text{min}} + \delta < \phi < \phi_{\text{max}} - \delta \\ \ln\left(\delta / \left(\phi_{\text{max}} - \phi\right)\right), & \phi_{\text{max}} - \delta \leq \phi < \phi_{\text{max}}, \end{cases} \tag{12}$$

where ϕ is the joint angle, ϕ_{min} and ϕ_{max} are the limits of that joint, and δ is the angular distance from each of the limits after which the limit function vanishes. Figure 3 shows an example of the limit function for $\delta = 45°$, $\phi_{\text{min}} = 90°$, and $\phi_{\text{max}} = 270°$.

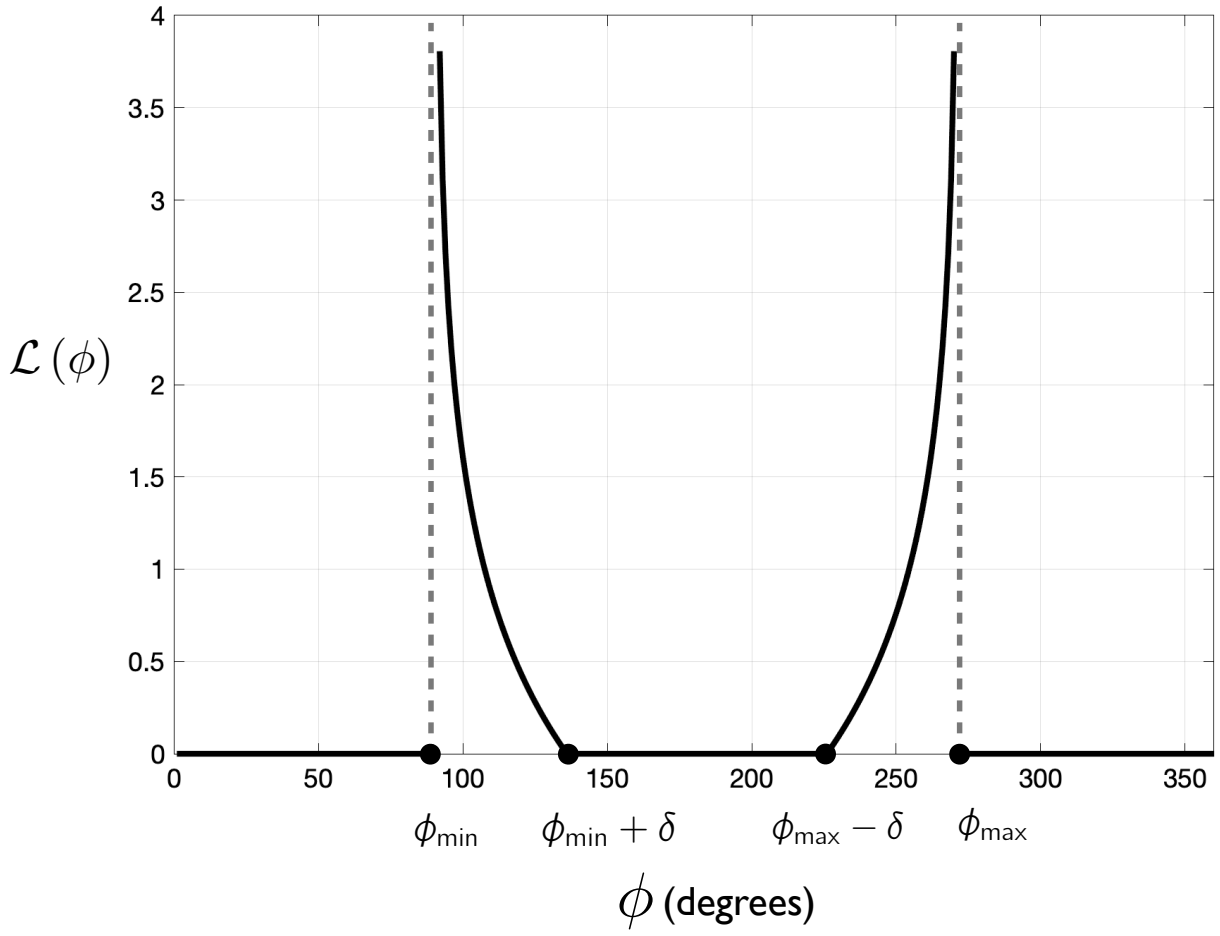


Figure 3: Limit function. One-dimensional plot of the limit function $\mathcal{L}$. For a given joint angle $\phi \in (\phi_{\text{min}}, \phi_{\text{max}})$, the penalty value increases as the joint angle approaches the limits.

References

[1] David E. Breen. Cost minimization for animated geometric models in computer graphics. *The Journal of Visualization and Computer Animation*, 8(4):201–220, 1997.