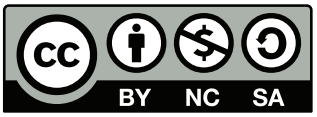
Haptic Feedback modelling for manipulator with 3DOF

Jakub Koban

23.01.2018

Department of Cybernetics and Robotics, Wrocaw University of Science and Technology
Class: Intermediate Project
Instructor: Witold Paluszyski Ph.D.



Abstract

The aim of this project is to explore possibilities of modelling a haptic feedback for 3DOF redundant manipulator in two dimensions using the Unity3D game engine. The behaviour of the manipulator (as real object) was simulated using inverse kinematics approach. The results illustrate feasibility, problems, and effectiveness of selected environment and mathematical methods.

1 Introduction

1.1 Assumptions

1. Design and create simple manipulator model in Unity3D.
2. Compute kinematics and inverse kinematics of such model, to mimic real object's behaviour.
3. Create as the simulation where manipulators behaviour could be controlled with respect to the force applied to its effector.
4. Create a visualization of important parameters (drag forces in joints, angles etc.)

The goal of this project in one sentence is: Control manipulator's joints drag forces in order to influence (slow down, stop) the movement of effector by analysing the force applied to this effector.

1.2 Haptic/Force feedback

Haptics, in general, refers to the branch of engineering dealing with tactile human-machine interfaces. Another similar term (often used alternatively) is force feedback - that term originates from control theory where force is used as input for the control system rather than for instance position. Haptics interfaces are often suitable for force feedback based control systems (because o compliance), which is why many devices are controlled using force feedback (e.g. surgical training machines [1], or game simulators).

1.3 Unity3D

Unity3D[2] is currently the most popular development platform using in building high-quality 3D and 2D games. One of the biggest advantages is that it supports all (25+ most popular platforms including Windows, Linux, Mac, Android, PlayStation, etc.) out of the box. Unity is using NVIDIA PhysX - real-time physics engine which provides realistic effects based on physics. It supports rigid body dynamics, soft body dynamics, even particles and cloth simulation. It was used eg [3] so it should be capable application. There are also a lot of tutorials that simplifies and speeds up the learning process.

2 Realization

3 Manipulator model

Creating graphical element in unity can be achieved in two ways: It can be composited of ready-to-use simple shapes (like a square, circle, hexagon) or exported from external graphical design tool (like Gimp, Adobe Photoshop or Blender in case of 3D). The links of manipulator were created using simple shapes. The effector was designed in Gimp. To enable physics interaction between models in the game engine one need to add a component called RigidBody2D, where important parameters like mass, drag, and constraints could be specified (more information could be found in [4]).

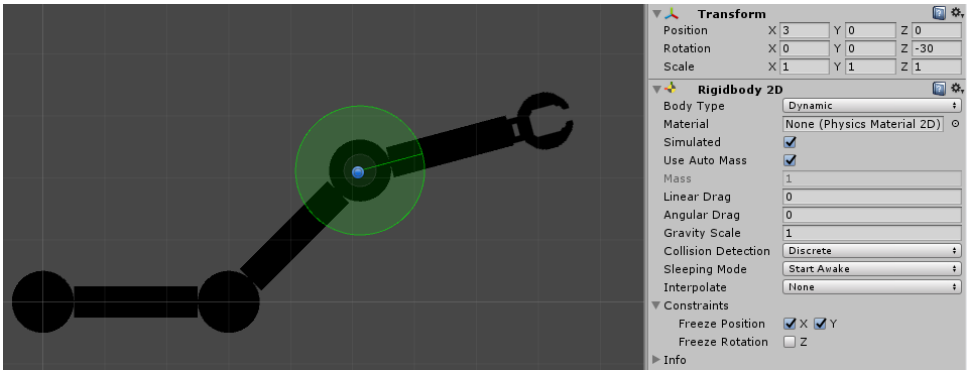


Figure 1: Manipulator model.

4 Kinematics and Inverse Kinematics

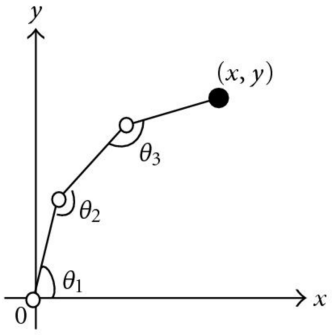


Figure 2: Three DOF planar manipulator simplified model

4.1 Analytic solution

Direct kinematics of manipulator model could be very easily derived analytically from figure 2

$$x = l_1 cos(\theta_1) + l_2 cos(\theta_1 + \theta_2) + l_3 cos(\theta_1 + \theta_2 + \theta_3) \tag{1}$$

$$y = l_1 sin(\theta_1) + l_2 sin(\theta_1 + \theta_2) + l_3 sin(\theta_1 + \theta_2 + \theta_3) \tag{2}$$

$$\phi = \theta_1 + \theta_2 + \theta_3 \tag{3}$$

Inverse kinematics are not that easy. They can be derived using some symbolic equation solver, however it results in equations that contains a lot of arcsin and arccos functions which are not numerically stable (see History and motivation section of [5]). Taking into account that facts after a lot of complicated transformations one can finally derive:

$$\theta_2 = atan2(sin(\theta_2), cos(\theta_2)) \tag{4}$$

$$\theta_1 = atan2((k_1 y_n - k2 x_n), (k_1 x_n - k2 y_n)) \tag{5}$$

$$\theta_3 = \phi - (\theta_1 + \theta_2) \tag{6}$$

where, $k_1 = l_1 + l_2 cos(\theta_2)$, $k2 = l_2 sin(\theta_2)$, $cos(\theta_2) = \frac{x^2+y^2-l_1^2-l_2^2}{2l_1 l_2}$, $sin(\theta_2) = \pm\sqrt{1 - cos^2(\theta_2)}$, $x_n = x - l_3 cos(\phi)$, $y_n = y - l_3 sin(\phi)$, l_1,l_2 and l_3 are lengths of manipulators links.

Having such implementation is ideal in theory, however in practice it also consists of several troubles. Mathematical representation above was firstly prototyped using Matlab. Such inverse kinematics implementation was working as expected. Unfortunately in Unity it does not because of its mathematical library imperfection. For instance computing sin and cos for very small numbers results in different results in Unity (using both C# Math library or Unity's MathF) and Matlab.

4.2 Jacobian method

One of many alternative methods to solve inverse kinematics problem is the Jacobian method. Very detailed description of this (and another Jacobian based) method could be found in [6]. Shortly Jacobian method iteratively computes an estimate of the distance to the desired position and minimizes the error. It is, in fact, an optimization problem, for which many good algorithms exist (e.g. Levenberg-Marquardt or Gradient Descent with Momentum). However in Unity dealing with matrices is also problematic (rotation represented with quaternions, so matrices are basically not needed), that is why simple gradient was implemented. Jacobian Inverse was substituted with transposition which also simplifies computations.

5 Dynamics

The necessary mathematics required to obtain manipulators dynamics (in Euler-Lagrange or Newton-Euler formalism) could be found in [7]. It is quite complicated and involves a lot of equations, so it is not presented explicitly here due to report's length constraints. Some crucial parameters (e.g. inertia matrices, rigid bodies velocities) are available in physX but unfortunately are not present in Unity3D (a huge part of PhysX API is hidden, because of optimization reasons). Due to lack of better linear algebra support, the haptic feedback realization was extremely simplified to affect the only effector, by changing angular drag parameter in last joints rigid body [8].

6 Visualization

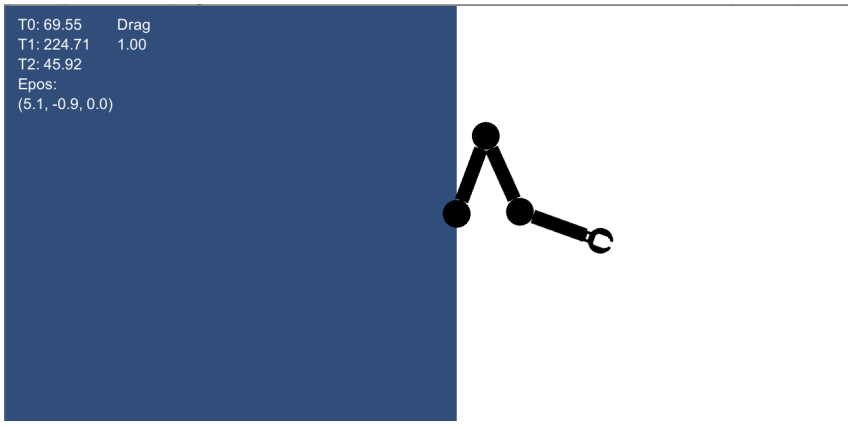


Figure 3: Visualization of manipulators parameters.

Figure 3 presents the visualization of most important manipulators parameters. T0-T2 represents angles in particular joints. Epos depicts effectors position, and Drag represents the actual value of angular drag in the last joint. Creating UI in Unity3D is very customizable and intuitive (more information could be found in [9]).

7 Tools licensing

1. Blender - open source.
2. Unity - closed software, free of charge for open source.
3. Matlab - closed software, free 30-day trial available for non-commerce.

8 Results

- Model of 3DOF manipulator was created. Its dynamical parameters were described using rigid bodies (defined in game engines).
- Kinematics and Inverse Kinematics was computer.
- Setting position of manipulators effector is possible in runtime (first two joints rotation using mouse, last joint orientation using up/down arrows.)
- Force interactions (haptic feedback) was simplified to the last joint because of game engines lack of precision and/or appropriate functions.
- Essential informations about manipulators were presented graphically.
- Jacobian method for solving inverse kinematics depends on minimization algorithm. Proposed algorithm sometimes hits local minima which is very annoying. Better algorithm (e.g. Levenberg-Marquardt) should fix that, however doing complicated computation in selected environment is not easy.
- Having analytical solution does not always mean that it could be used directly in practice (atan2 should be used instead of arcsin and arccos - expressing basic formula in more comfortable form may be, as here very difficult).
- Unity3D is very powerful tool however it is not capable of doing precise mathematical computations, and its physics engine is computing everything approximately not precisely.
- There are not many scientific publications about haptic/force feedback methods - it is often necessary to search for informations in papers of different (but similar) subjects.

References

[1] W. Rozenblit M. Hong. A haptics guidance system for computer-assisted surgical training using virtual fixtures. *2016 IEEE International Conference on Systems, Man, and Cybernetics*, 2016.

[2] *https://unity3d.com/unity.*

[3] Zhonghua Lu A. Maciel, T. Halic. Using the physx engine for physics-based virtual surgery with force feedback. *Int J Med Robot 5(3): 341- 353*, 2009.

[4] *https://docs.unity3d.com/ScriptReference/Rigidbody2D.html.*

[5] *https://en.wikipedia.org/wiki/Atan2.*

[6] M. Opaka I. Dulba. A comparison of jacobian-based method of inverse kinematics for serial robot manipulators. *nt. J. Appl. Math. Comput. Sci.*, 2013.

[7] A. Ghanbari SMRS. Noorani. Explicit dynamic formulation for n-r planar manipulators with frictional interaction between end-effector and environment. *International Journal of Advanced Robotic Systems*, 2011.

[8] *https://docs.unity3d.com/ScriptReference/Rigidbody2D-angularDrag.html.*

[9] *https://unity3d.com/learn/tutorials/s/user-interface-ui.*