A motor control model

Thomas Beucher

May 21, 2015

Abstract

Two basic phenomena interact in the way the speed of our reaching movements is determined. First, we tend to reach faster a target that looks more rewarding, despite the additional muscular cost of a faster movement. Second, when we need to be more precise, our movement takes more time. So far, these two phenomena have been studied in isolation despite their obvious interdependency. In particular, two recent computational models of motor control address the first phenomenon. They explain the emergence of the time of movement as resulting from a cost-benefit trade-off arising from the summation of a temporally discounted reward and a cost that increases for faster movements. However, these models do not account for the second phenomenon, i.e. the dependency between movement time and precision requirements, resulting in a speed-accuracy trade-off and formally expressed by Fitts' law. Another model addresses the role of this speed-accuracy trade-off in determining movement time, but does not take the cost of movement into account.

In this paper, we propose a framework that unifies the cost-benefit trade-off and the speed-accuracy trade-off to explain movement properties related to time. With respect to the cost-benefit trade-off models, precision constraints are incorporated through the derivation of a new optimization criterion that considers probabilistic reaching of a rewarding target that may be missed if the motion is too fast.

Using this computational model, we investigate the more global trade-off arising from the interactions between movement time, cost and accuracy. We show that this model accounts for Fitts' law and for other well-established results in the motor control literature.

Chapter 1

Introduction

This report is about my internship at ISIR (Institut des Systmes Intelligents et de Robotique) covering a period of five months from February to July 2015.

The Institute for Intelligent Systems and Robotics (ISIR) is a multidisciplinary research laboratory that brings together researchers and academics from different disciplines of Engineering Sciences and Information and the Life Sciences. It is based in Paris France.

1.1 State of the art

There has been a recent progress in motor control research on understanding how the time of a reaching movement is chosen. In particular, two recent models from Shadmehr et al. [**?**] and Rigoux&Guigon [**?**] proposed an optimization criterion that involves a trade-off between the muscular effort and the subjective value of getting the reward, hence a cost-benefit trade-off (CBT). On one hand, reaching a target faster requires a larger muscular effort (refs?). On the other hand, the subjective value of reaching a target decreases as the time needed to reach the target is increased (refs?). As a result, the net expected return consisting of the subjective value minus the muscular effort is optimal for a certain time, as illustrated in Fig. 1.1(A).

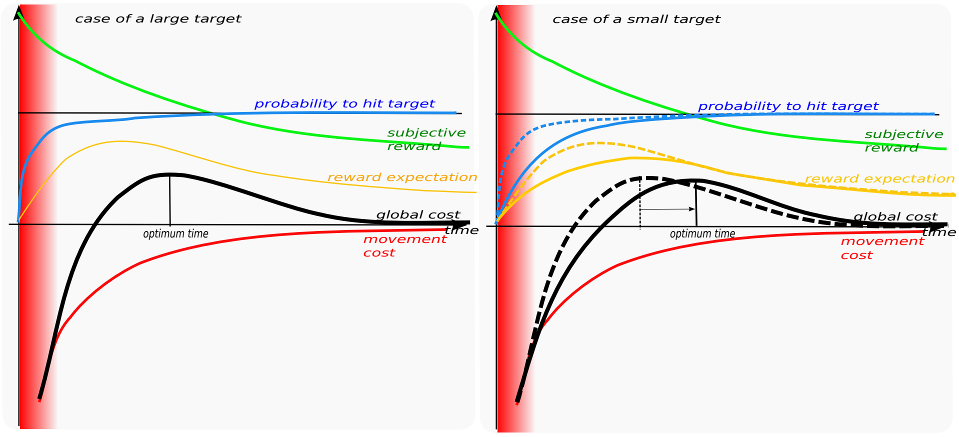

Figure 1.1: Influence of movement time on cost related quantities. Green: subjective utility of hitting the target; red: muscular energy cost; black: global cost versus reward trade-off. The red area denotes infeasible short times; blue: probability to hit the target; orange:reward expectation (subjective reward times probability). A: Sketch of the models in [**?**] and [**?**]. The subjective utility of hitting the reward decreases over time as one is less interested in gains that will occur in a distant future than at the present time. Hitting is less and less costly in terms of efforts as the movement is performed more slowly. The expected gain, resulting from the sum of the subjective reward and the (negative) cost reaches a maximum for a certain time. When the gain is negative (outside the useful interval), one should not move. B: Sketch of the presented model. In the case of a larger target, the hitting probability is higher for faster movements (solid lines) than for a smaller target (dashed lines). As a result, the maximum of the reward expectation is shifted towards longer time for smaller targets, and the optimum movement time is also longer for smaller targets.

However, these models do not account directly for basic facts about the relation between movement difficulty and movement duration as captured more than fifty years ago by Fitts' law [?]. According to this law, the smaller a target, the slower the reaching movement. This is well explained by the so-called *speed-accuracy trade-off* (SAT) stating that, the faster a movement, the less accurate it is, hence the higher the probability to miss the target. So a subject reaching too fast may not get the subjective value associated to reaching and should slow down.

In contrast with the models of [?] and [?], the model of Dean [?] takes the SAT into account. The key difference with respect to [?] and [?] is that, instead of maximizing a reward, this model maximizes a *reward expectation*, i.e. the reward times the probability to get it.

However, the model proposed in [?] is an abstract model of movement time selection that looks for an optimal trade-off between an externally decayed reward and a SAT that relates the probability of missing to movement time. As such, it does not account for movement execution, neither for the choice of a motor trajectory and its impact on the cost of movement. The model does not explain Fitts' law, it rather incorporates its consequences into an abstract model of the SAT that is fitted to experimental data. The mathematical design of the model is based on several simplifying assumptions and it predicts optimal movement times that are systematically shorter than those observed with subjects. The authors of [?] discuss that this may result from the fact that the model does not take the cost of movement into account.

In the paper of Olivier Sigaud and Kevin Monfray, they show that the models of Shadmehr [?] and Rigoux [?] as well as the model of Dean [?] can be unified into a model that solves the difficulties faced by these previous models.

This unification is simply implemented by including sensory and motor noise into the optimal control model proposed in [?], shifting from a deterministic account of the movement to a stochastic one, in line with the models of [?, ?, ?, ?, ?].

As a matter of fact, in the models of [?] and [?], the target is given as a single point and the movement is considered as always reaching it, irrespective of the size of the target. In order to fully account for Fitts' law, one must consider the intrinsic dispersion of reaching movements towards a target and the effect of sensory and muscular noise on this dispersion (e.g. [?], see [?] for a review), which is not the case of the models of [?] and [?].

As a result, the reward and muscular activation terms in the optimization criterion proposed in [?] are replaced by reward and cost expectation terms. Considering expectation is a way to account for the fact that, in case of a miss, one would not get the reward, so the global outcome of the movement would only consist of its incurred cost.

1.2 Previous work

I used a code, written by Didier Marin (a PhD student at ISIR), which implement a model based on optimal control called NOPS but it is very time consuming. To reduce the computation time, Olivier Sigaud decided to try to learn the NOPS controller using a regression algorithm. So I generated trajectories using the NOPS controller then I used the regression algorithm called RBFN(Radial Basis Function Networks) to learn the new controller from these trajectories. The new controller could then be optimized with a stochastic optimization method called CMAES(Covariance Matrix Adaptation Evolution Strategy).

1.3 My job

First, I have to implement the model of arm and the regression algorithm to learn the new controller from the trajectories generated with the NOPS controller (implemented by Didier Marin in C++ and in Java by Kevin Monfray). Secondly, I must use the CMAES algorithm to optimize RBFN controller for different sizes of target. Finally, generate various relevant curves and compare them with the curves obtained by a Slovenian colleague who experiment on humans around the same issue.

Chapter 2

Material and methods

2.1 Arm model

The plant is a two degrees-of-freedom (dofs) planar arm controlled by 6 muscles, illustrated in Fig. 2.1. There are several such models in the literature. The model described in [?] lies in the vertical plane so it takes the gravity force into account. Most other models are defined in the saggital plane and ignore gravity effects. They all combine a simple two dofs planar rigid-body dynamics model with a muscular actuation model. The differences between models mostly lie in the latter component.

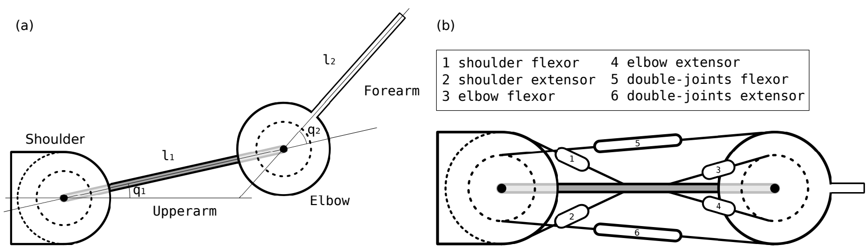

Figure 2.1: Arm model. (a) Schematic view of the arm mechanics. (b) Schematic view of the muscular actuation of the arm, where each number represents a muscle whose name is in the box.

Table 4.1 in Appendix 4.1 reminds the nomenclature of all the parameters and variables of the arm model.

2.1.1 Arm parameters

All parameters of the arm are defined in the file *setupArmParameters* and implemented in the class *ArmParameters*. This class defines the following functions:

readSetupFile : Reads the setup file.

massMatrix : Defines the inertia matrix parameters.

BMatrix : Defines the damping matrix $\mathbf{B}$, with $\mathbf{B} = \begin{bmatrix} .05 & .025 \\ .025 & .05 \end{bmatrix} \dot{q}$.

AMatrix : Defines the moment arm matrix A.

$$\begin{aligned} \mathbf{A}^\top &= \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 & a_{10} & a_{11} & a_{12} \end{bmatrix} \\ &= \begin{bmatrix} .04 & -.04 & 0 & 0 & .028 & -.035 \\ 0 & 0 & .025 & -.025 & .028 & -0.35 \end{bmatrix} \end{aligned}$$

All the arm parameters values are summarized in Table 4.2 in Appendix 4.1.

2.1.2 Muscles parameters

All muscles parameters are defined in the file *setupMusclesParameters* and implemented in the class *MusclesParameters*. This class defines the following functions:

fmaxMatrix : Defines the matrix of the maximum force exerted by each muscle.

$$\begin{aligned} \mathbf{f_{max}} &= \begin{pmatrix} f_{\max 1} & 0 & 0 & 0 & 0 & 0 \\ 0 & f_{\max 2} & 0 & 0 & 0 & 0 \\ 0 & 0 & f_{\max 3} & 0 & 0 & \\ 0 & 0 & 0 & f_{\max 4} & 0 & 0 \\ 0 & 0 & 0 & 0 & f_{\max 5} & 0 \\ 0 & 0 & 0 & 0 & 0 & f_{\max 6} \end{pmatrix} \\ &= \begin{pmatrix} 700 & 0 & 0 & 0 & 0 & 0 \\ 0 & 382 & 0 & 0 & 0 & 0 \\ 0 & 0 & 572 & 0 & 0 & 0 \\ 0 & 0 & 0 & 445 & 0 & 0 \\ 0 & 0 & 0 & 0 & 159 & 0 \\ 0 & 0 & 0 & 0 & 0 & 318 \end{pmatrix} \end{aligned}$$

activationVectorInit : Initializes the muscular activation vector. (Create the vector and initializes to zero)

activationVectorUse : Builds the muscular activation vector given its 6 components.

All the muscles parameters values are summarized in Table 3 in Appendix 4.1.

2.1.3 Rigid-body dynamics

The rigid-body dynamics equation of a mechanical system is:

$$\ddot{q} = \mathbf{M}(q)^{-1}(\tau - \mathbf{C}(q, \dot{q}) - \mathbf{g}(q) - \mathbf{B}\dot{q}) \tag{2.1}$$

where $\mathbf{q}$ is the current articular position, $\dot{q}$ the current articular speed, $\ddot{q}$ the current articular acceleration, $\mathbf{M}$ the inertia matrix, $\mathbf{C}$ the Coriolis force vector, $\boldsymbol{\tau}$ the segments torque, $\mathbf{g}$ the gravity force vector and $\mathbf{B}$ a damping term that contains all unmodelled effects. Here, $\mathbf{g}$ is ignored since the arm is working in the sagittal plane. All angles are expressed in radians. We can compute the inertia matrix as: $\mathbf{M} = \begin{bmatrix} k_1 + 2k_2\cos(q_2) & k_3 + k_2\cos(q_2) \\ k_3 + k_2\cos(q_2) & k_3 \end{bmatrix}$, with $k_1 = d_1 + d_2 + m_2 l_1^2,\ k_2 = m_2 l_1 s_2,\ k_3 = d_2$ where d_i, m_i, l_i and s_i are parameters of the arm previously defined in Section 2.1.1.

The Coriolis force vector is given by

$$\mathbf{C} = \begin{bmatrix} -\dot{q_2}(2\dot{q_1} + \dot{q_2})k_2\sin(q_2) \\ \dot{q_1}^2 k_2 \sin(q_2) \end{bmatrix}.$$

The computation of the torque τ exerted on the system given an input muscular actuation $\mathbf{u}$ is explained in the section 2.1.4.

Equation 2.1 is implemented in the class *ArmDynamics* line 57:

```
57        ddotq = np.dot(Minv,(Gamma - C - np.dot(armP.B,
              dotq)))
```

where *np* refere to the numpy library in python. We also find in this class all elements of Equation 2.1:

```
44        #Inertia matrix
45        M = np.array([[armP.k1+2*armP.k2*math.cos(q[1,0]),
              armP.k3+armP.k2*math.cos(q[1,0])],[armP.k3+armP
              .k2*math.cos(q[1,0]),armP.k3]])
46        #Coriolis force vector
47        C = np.array([[-dotq[1,0]*(2*dotq[0,0]+dotq[1,0])*
              armP.k2*math.sin(q[1,0])],[(dotq[0,0]**2)*armP.
              k2*math.sin(q[1,0])]])
48        #inversion of M
49        Minv = np.linalg.inv(M)
50        #torque term
51        Q = np.diag([q[0,0], q[0,0], q[1,0], q[1,0], q
              [0,0], q[0,0]])
52        #the commented version uses a non null stiffness
              for the muscles
53        #Gamma = np.dot((np.dot(armP.At, musclesP.fmax)-np
              .dot(musclesP.Kraid, Q)), U)
```

```
Gamma = np.dot((np.dot(armP.At, musclesP.fmax)-np.dot(musclesP.Knulle, Q)), U)
```

2.1.4 Muscular actuation

Our muscular actuation model is taken from [?] (pp. 356-357) through [?]. It is a simplified version of the one described in [?] in the sense that it uses a constant moment arm matrix $\mathbf{A}$ whereas [?] is computing this matrix as a function of the state of the arm.

Finally, given an action $\mathbf{u}$ corresponding to a raw muscular activation as output of the controller, the muscular activation is augmented with Gaussian noise using $\tilde{\mathbf{u}} = \log(\exp(\kappa \times \mathbf{u}_t \times (1 + \mathcal{N}(0, \mathbf{I}\sigma_u^2))) + 1)/\kappa$, where $\times$ refers to the element-wise multiplication, $\mathbf{I}$ is a 6×6 identity matrix. and $\kappa = 25$ is the Heaviside filter parameter, and the input torque is computed as $\boldsymbol{\tau} = \mathbf{A}^\top(\mathbf{f_{max}} \times \tilde{\mathbf{u}})$.

Chapter 3

Study

Chapter 4

Appendix

4.1 Nomenclature of arm parameters

Table 4.1: Parameters of the arm model.

m_i	mass of segment i (kg)
l_i	length of segment i (m)
s_i	inertia of segment i ($kg.m^2$)
d_i	distance from the center of segment i to its center of mass (m)
κ	Heaviside filter parameter
$\mathbf{A}$	moment arm matrix ($\in \mathbb{R}^{6\times 2}$)
$\mathbf{f_{max}}$	maximum muscular tension ($\in \mathbb{R}^6$)
$\mathbf{M}$	inertia matrix ($\in \mathbb{R}^{2\times 2}$)
$\mathbf{C}$	Coriolis force ($N.m \in \mathbb{R}^2$)
τ	segments torque ($N.m \in \mathbb{R}^2$)
$\mathbf{B}$	damping term ($N.m \in \mathbb{R}^2$)
$\mathbf{u}$	raw muscular activation (action) ($\in [0,1]^6$)
σ_u^2	multiplicative muscular noise ($\in [0,1]^6$)
$\tilde{u}$	filtered noisy muscular activation ($\in [0,1]^6$)
$\mathbf{q}^*$	target articular position ($rad \in [0, 2\pi[^2$)
$\mathbf{q}$	current articular position ($rad \in [0, 2\pi[^2$)
$\dot{q}$	current articular speed ($rad.s^{-1}$)
$\ddot{q}$	current articular acceleration ($rad.s^{-2}$)

Table 4.2: Parameters of the arm.

$\mathbf{l_1}$	arm length (m)	0.3
$\mathbf{l_2}$	forearm length (m)	0.35
$\mathbf{m_1}$	arm mass (kg)	1.4
$\mathbf{m_2}$	forearm mass (kg)	1.1
$\mathbf{s_1}$	arm inertia ($kg.m^2$)	0.11
$\mathbf{s_2}$	forearm inertia ($kg.m^2$)	0.16
$\mathbf{d_1}$	distance from the center of segment 1 to its center of mass (m)	0.025
$\mathbf{d_2}$	distance from the center of segment 2 to its center of mass (m)	0.045
$\mathbf{k_6}$	damping term	0.05
$\mathbf{k_7}$	damping term	0.025
$\mathbf{k_8}$	damping term	0.025
$\mathbf{k_9}$	damping term	0.05
$\mathbf{a_1}$	moment arm matrix	0.04
$\mathbf{a_2}$	moment arm matrix	-0.04
$\mathbf{a_3}$	moment arm matrix	0.0
$\mathbf{a_4}$	moment arm matrix	0.0
$\mathbf{a_5}$	moment arm matrix	0.028
$\mathbf{a_6}$	moment arm matrix	-0.035
$\mathbf{a_7}$	moment arm matrix	0.0
$\mathbf{a_8}$	moment arm matrix	0.0
$\mathbf{a_9}$	moment arm matrix	0.025
$\mathbf{a_{10}}$	moment arm matrix	-0.025
$\mathbf{a_{11}}$	moment arm matrix	0.028
$\mathbf{a_{12}}$	moment arm matrix	-0.035

Table 4.3: Parameters of the muscles.

$\mathbf{f_{max1}}$	Maximum force exerted by the shoulder flexor	700
$\mathbf{f_{max2}}$	Maximum force exerted by the shoulder extensor	382
$\mathbf{f_{max3}}$	Maximum force exerted by the elbow flexor	572
$\mathbf{f_{max4}}$	Maximum force exerted by the elbow extensor	445
$\mathbf{f_{max5}}$	Maximum force exerted by the double-joints flexor	159
$\mathbf{f_{max6}}$	Maximum force exerted by the double-joints extensor	318