A Parallel Approach to the Simulation of Complex Movements

Paolo FRANCHI, Vittorio SANGUINETI, Giancarlo SUCCI, Gianluca VERCELLI

DIST-Università di Genova via Opera Pia 11a, 1-16145 Genova, Italy

Abstract. A simulation environment is presented aimed at representing and simulating complex kinematic structures, based on a gradient descent process, operating on a sort of elastic potential field defined over the set of involved structures. At higher level a trajectory formation tool, also based on the minimization of a global potential energy function and on cooperating units, is responsible for organizing and driving the evolution of the represented system. The whole model is inherently parallel and distributed; an implementation on a MIMD transputer-based architecture is presented, as well as some simulation examples. A linguistic tool, M-Language, is provided to interact with this environment, in order to specify complex mechanical structures in terms of primitive mechanical units as well as complex movements in terms of basic actions, timed through sequential/parallel constructs.

1 Introduction

Human movements are very difficult to describe and to simulate, because of the structural complexity of the musculoskeletal system and of the complex timing structure underlying virtually all real world motor tasks.

One main feature of human and humanoid musculoskeletal systems is their structural redundancy, that is, they involve a large number of degrees of freedom (dofs), so that, for instance, a particular hand position can be achieved by several different joint configurations and by several different muscular activations combinatorially. Redundancy is basically responsible for human dexterity, that is, the ability to choose different strategies, dependent of environmental constraints, to perform a particular motor task. Advanced robotic applications, aimed to reproduce human dexterity, also involve complex (possibly multiple and redundant) manipulators, and complex robot plans. The problem of simulating the movement of complex structures is not trivial and can be examined from a biological point of view. In this framework, motors planning is usually addressed as synergy formation and can be looked at as an information processing task, translating a concise and abstract motor plan into an array of time functions, one for each muscular actuator, depending on structural and environmental constraints. The synergy formation mechanism is known to be independent of the function of dofs, either kinematic or muscular, moreover, to consider redundancy as an exception can be shown misleading, because even the simplest human movement can virtually involve all the muscles and the joints in the body.

The remark [1] that muscles should not be considered as force or displacement generators, but, from a mechanical point of view, behave as non-elastic elements (that is, as elastic springs), thus being able to store elastic energy, has led to the idea that a postural equal to a minimum total potential energy configuration for the whole musculoskeletal system and is completely determined by the particular set of neural commands that are applied to each muscle; motor commands from motoneurons "select" a particular length-tension source for each muscle. In more formal terms, a posture corresponds to the condition of zero torque for each joint, i.e.:

$$\tau(q, a) = 0$$

(1)

where $q$ is the neural activity vector; in other words the equilibrium configuration $q^*$ minimizes the quantity:

$$E_q(q, a) = \int T_q(q, a) dq$$

(2)

Moreover, there is some experimental evidence [2] that movements are performed through a slight anheuristic of the neural activity of each muscle, in such a way that each intermediate body configuration can also be considered as a minimum $\epsilon$-energy equilibrium point; in other words, a movement is a flow of postures and equal variation in a movement invariance. From the theorem of implicit functions we can obtain that the equilibrium configuration $q^*(a)$ can be uniquely determined by the muscle activation $a$, a result known as Equilibrium Point (EP) hypothesis. An important prediction of the EP hypothesis is that the process of generation of movements is only influenced by the "static" components of the involved mechanical structures (muscles and ligaments), thus only static properties need to be modeled, at least to simulate "slow" movements.

Given a desired final displacement of any point of the structure (end-effector), expressed in environment coordinates, the corresponding displacement of body configuration can be found by locally inverting through a gradient descent mechanism, the direct kinematic transformation between body configuration $q$ and end-effector position $x$, so that internal constraints, like joint limits and the balance of ligament and muscular forces are satisfied; this approach was first proposed by [2] and is known as the passive muscle paradigm (PMP). Models based on PMP, also known as EP models, can be seen as computational tools aimed at solving the so-called inverse kinematics (IK) problem (the problem of finding the joint trajectory corresponding to a given end-effector trajectory), whereas the set of internal constraints defines a particular pseudo-inverse of the Jacobian of the direct kinematic transformation. A similar approach was proposed as a general algorithm to solve IK problem for manipulators without explicitly inverting the Jacobian matrix [4].

Another key property of human movements, also desirable in dexterous robust manipulators, which should also be accounted for a simulation framework, is their "smooth" appearance, yielding a bell-shaped velocity profile [3] in the simplest case of reaching movements; EP models are not concerned with the temporal and spatial structure of the trajectories in environment space, which should be specified by some higher level trajectory planner.

The generation of motor commands, corresponding in the biological case to muscle activations (the so-called inverse musculoskeletal (IM) problem, also affected by redundancy at musculoskeletal level), cannot be easily performed by EP models because they are based on considering both muscles and ligaments as passive strings, thus hypothesizing that muscular activations are kept constant during a single step. In fact, a further computational mechanism should be provided, responsible for generating the muscle activation increments corresponding to the displacements in body configuration, computed by the EP model.
Therefore, a complete computational model for movements generation, based on the EP paradigm, should involve at least three stages (see figure 1), accounting for trajectory generation and for the solution of, respectively, the IK problem and the IM problem.

### The Simulation Environment

The proposed simulation environment is aimed to manage the complex mechanical structures of musculoskeletal systems and of robotic manipulators, as well as to deal with complex movements: inspired by the EP concept, its main feature is that both mechanical structures and movements can be represented in a distributed way, thus making possible a parallel implementation. It involves two components: M-Language and M-Architecture.

- **M-Language** is aimed at the linguistic description of the constrained kinematic structures and the (possibly concurrent) motor actions. The underlying philosophy is related to the object oriented paradigm: the user is able to describe a complex structure as well as a complex movement in terms of simple primitive mechanical units and elementary motor activities.

- **M-Architecture** is the corresponding computational tool for the generation of motor commands compatible with the imposed structural or task-related constraints, involving two components: the Structure Level and the Task Level.

### M-Architecture

#### 3.1 The Structure Level

It is responsible for processing the description of the involved models supplied by an M-Language script and uses an EP-based simulation tool, Motor relaxation Network (M-Net), which can be seen as a representation of the task-independent component of motor knowledge.

A M-Net model is made by Units, representing the skeletal system, the muscular actuators and the other passive constraints (ligaments, joint limits). A complex musculoskeletal system can be represented as a semantically organized graph of the following entities:

- **M-Units** (Muscle Units) take into account the effect of muscular actuators, described by a continuous, in general non-linear function:
  \[ f_{u} = f(u, \theta, \alpha) \]
  corresponding to the physiological length-tension curve [6].

#### 3.2 The Task Level

In the proposed approach, M-Net is the task-independent component of motor knowledge regarding goal selection and timing.

This requires a planning mechanism operating at a more abstract level; the Task Level is responsible for the generation of the sequence of trajectory via-points driving the M-Net level, starting from a linguistic description of the desired motor task.

The starting idea is that basic motor tasks can be simply described in terms of a target position and any motor task can be expressed as a composition of such basic tasks, so the generation of synergy vectors generation is a (possibly constrained) vector interpolation problem in environment space. Such an interpolation takes into account the fact that the target can be accomplished by a potential function-based mechanism; one can think of M-Net models, each attracted by its target by an elastic force and possibly repelled...
\[ F = \kappa (q' - q_0) \]  
(6)

The trajectory via-point moves during the simulation, due to an higher level planner level.

S-Units (Skeletal Units) are to mechanical links (bones in the biological case) and are modeled as rigid bodies, so that their status can be simply described through the position and the orientation of a local reference frame.

J-Units (Joint Units) represent the rotational joints of the kinematic structure; they are the only dynamic objects in the model, acting as computational units; for each unit, the status dynamics is defined by the following:

\[ q = \eta \left[ f_M^r(t) + f_c^r(t) + f_j^r(t) \right] \]  
(7)

where the input variables are M-Units and L-Units forces, \( f_M^r \) and \( f_c^r \) are J-Units torques, \( r \) and A-Units forces, \( F \). Equation 7 defines a gradient descent dynamics, minimizing the global elastic potential energy, with the constraint introduced by the A-Unit component, corresponding to a sort of penalty function.

By means of H-Net, a complex mechanical structure can be represented as a tree structure in which S-Units are the nodes, whereas the J-Units are the arcs. S-Units are also connected one other by M-Units and L-Units, accounting for the elastic structural components (see figure 2).

A displacement from the equilibrium configuration, induced by any A-Unit, originates a sort of relaxation in the structure, that tends to stabilize itself in a new equilibrium configuration; in other words, the desired trajectory via-point behaves as an obstacle; in this case, equilibrium configuration corresponds to having reached each target.

In this sense, the natural way to describe complex motor tasks is to use concurrent procedures, timed through the use of temporal statements. The structure of a complex action, possibly involving many models, is better solved by a distributed approach: the implicit parallelism between the atomic motor commands, the need of an accomodation coordination are only two aspects of the computational complexity when dealing with real scenarios. Programming and planning of movements in a classical way \( a, b \), in fact, considered separated, off-line activities. But it is useless to consider sequent ally these two activities. Planning is, in our view, the result of a simulation of the problem using a metaphor of the real model. For example, if we consider a grasping task performed by a multi-fingered robotic hand, the set of atomic activities related to the reaching, size-shape and closure phases can be serialised, as in the classical robot programming a sapproach.

It is out of doubt that we can obtain a more realistic ( anthropomorphized ) performance if the three different phases are temporally overlapped using a parallel guarded mechanism. Moreover, in order to grasp the given object in a stable manner, the parallel activation of motor commands for each finger must be synchronized with the sensorial and stability processes. Having a single syntax, and a precise semantics, as we have done with M-Language, we can build a distributed planning/programming environment able to solve the general problem in terms of sequential/parallel activation of procedure. The implicit parallelism of complex task is a source of complexity when using classical architecture. Our system has been designed as a coarse-grained parallel architecture: we take advantages from the timed, compositional structure of the concurrent procedures, and using a proper set of temporal statements, we may schedule and synchronize the whole concurrent simulation. On the other side, it is not efficient to distribute everything all the units over the available computing platform (we think to a general-purpose MIMD machine): the communication overhead between units must be clearly low with respect to that one of the computational part.

We called the concurrent activations of basic motor actions the task level ("P-Level") of our system. In this level the motor tasks to be performed are syntactically described in terms of goals and activation instances; all the kinematic constraints are left to the structure level of the M-architecture, which are task-independent and are responsible of the generation of streams of kinematic expectations.

It should be noted that a powerful characteristic of this approach to trajectory formation is its compositional/parallel/sequential/simultaneous nature. Multiple units of each kind can operate concurrently in order to express multiple goals inherent in the same motor task. The task level can be represented as a collection of Planning Units (P-Units), that is a Planning Execution Network (P-Net), which is superimposed to a set of M-Net models and generates the synergy vectors.

For each P-Unit the system dynamics is described by:

\[ f_{P}^{\text{in}} = f - GO(\text{Input}) \]  
(8)

GO(\text{Input}) is a scalar, monotonic function and is responsible for movement initiation and duration and for bell-shaped velocity profile. Such a mechanism is very similar to \( VITE \) (Vector Integration To Endpoint), a connectionist model proposed in [1], [8] for motor control in musclr space.
4 M-Language

M-Language is an actor-based object-oriented programming language, able to combine analog simulators with traditional procedures in order to deal with external sensorial systems. Using M-Language it is possible to define a "world model" in terms of models (kinematic structures), actors and actions. The first part of a M-Language script is the configuration of the modeled structure; it contains the description section, which is composed by declaration of all the units and their physical and geometrical relevant properties. The second part concerns the distributions of the models into a graph of composing elements.

Models are objects composed of M-Net, with a specific role of passive entities (in the sense of PMP) that react to changes of the world. Models could be simple or compound. Each model corresponds to a particular M-Net, referenced by its name; it allows the creation of multiple instances of the same generalised model, simply using a different name for each instance. Several properties have to be set for each kind of unit: name, position, mass, center of gravity, stiffness, rest length, actual length, max force, max length, actual extension and relationship with connecting units.

Actions are objects that act as active entities in the world model. They can be attractors for the relaxation phase of M-Net or motions (active timed procedures) of actors. Several control constructs (special timed control structures) are available to allow the user to define the motor task to be accomplished.

Events are objects that change the state of the world model. They can be the output of generalized sensing or some kind of decision making.

The world model evolves through asynchronous relaxation phases of Models, guided by attractors and concurrent execution of motions; all the actions are timed by events and temporal statements.

The last part of the program file concerns the distribution of the models into a graph of computational entities (processors). We assume that a collection of processors is available, in conjunction with a Communicating Sequential Processes (CSP) mechanism [9]. This part is responsible for the definition of the parallel structure of a compound model. By definition, each model/block is assumed to be assigned to a single processor. A compound model is a connected graph of single models and motions (each one assigned to a particular processor) communicating by means of channels (bidirectional queues of messages).

For example, if we want to build a distributed model of a three-fingered hand that performs a grasping task, we might collect together previously defined models, stored in a object-oriented hierarchy of classes, we can use three instances of the OneFinger model (see below), one instance of the Palm model and one instance of the GraspCoordinator actor.

Note that this model logically indicates how to distribute the computational entities and how they communicate among each other, without the need to know whether a true parallel distributed hardware platform is available or not. If only a traditional architecture is available, then all the models become intercommunicating processes on a single processor.

4.1 Temporal Statements for Action Planning

The syntax of M-Language is derived from the NEM++ Language [10]; in this section will be described only the relevant differences with respect to this language.

It is possible to define the temporal relations among the planned tasks, by means of the time statements. Each time statement controls a block of actions either timed or non-timed. Once specified the semantics of the basic constructs, their combination and nesting describe an uniquely determined semantics.

We can define the temporal statements, that are used to specify the temporal order in which tasks have to be executed. The two main statements are sequence and together.

sequence is used to state that the tasks composing the block have to be sequentially executed. Each of the composing blocks is executed only after the completion of the preceding one; the sequence construct ends when the last composing task is completed.

together is the construct to express a concurrent execution of tasks. It is alive until at least one of the composing constructs is active. When a together block is activated, the first step of all the composing constructs is executed, and the computation proceeds by executing a step of all the concurrent active tasks.

Some basic actions are also predefined:

reach causes the planner to generate a straight trajectory for the end-effector to reach the specified point, either a fixed object or a moving one, such as another robot end-effector;

repel allows the creation of a trajectory to drive the end-effector at a given distance from an attractor;

take is used to connect the end-effector to an object; it cannot be considered an atomic task, but it is composed of a phase of reaching before the grasping itself;

release is the opposite of take.

5 Exploiting the Intrinsic Parallelism

M-Net philosophy is well suited for a parallel and distributed implementation, like a transputer one [11] since it has a low communication band and most of the computation can be executed independently. We have tested our architecture on several simples examples of kinematic structures (OneFinger, TwoFingers, Twolegs, ThreeFingeredHand, FiveFingeredHand); in figure 5 the simple graphic output of the ThreeFingeredHand is shown.

For the FiveFingeredHand model the idea is to use 5 of the 40 nodes of our Meko Computing Surface®; the nodes are divided in 4 classes according to their duties:

- Finger,
- Bridge,
- X Server,
- Planner.
The FiveFingeredLand nodes perform the simulation of 5 fingers independently of each other, getting strategy to reach the target from the Planner and sending the result of their movements to the X Server. The two bridges are used to enhance the performance of the system, as it is explained here below. The X Server provides an HTTP based interface of the system for the user. The duty of the Planner is to coordinate the actions of the fingers, updating the status of the A-units, it receives from all the fingers their position and it send them back the new positions of the attractors; in order to compute such positions it check whether the target has already been reached.

Figure 3 explains the topology of the system. Three of the five fingers are connected to the X Server and to the Planner directly, while two of them use two “Bridge” nodes, which have the duty of minimizing the communication overhead and are imposed by the fact that each transporter have only four links.

Figure 4 presents the design we are taking for attacking the problem of coordinating complex M-Net models. We have devised the M-Cell: it is the set of computational entities aimed to simulate a complex but well structured M-Net model, as the FiveFingeredLand, which does not act any more alone but in a more elaborate framework. The structure of the cell is close to the one explained in the previous paragraph, despite its semantics is rather different.

The M-Cell has 4 links with the world; three of them can be used for computational activities while the forth one can be connected to an X server. However since the transporter is a very flexible architecture their behavior is not strictly fixed and they can be used for different purposes.

The M-Cell for the FiveFingeredLand is composed of 12 nodes of the transporter: 5 fingers, 4 bridges, 1 planner, 1 X server connector and 1 gateway. The purpose of the gateway is to obtain a high degree of connectivity for the cell: the 4 links assure a kind of homogeneity with the transporter environment and offer opportunities for various cells topologies. There isn’t any more a X server in the cell, since the cells are more than one; instead there is an X server connector whose duty is to provide fast and efficient communications with the X server. The desire of keep the regularity of the system and to achieve a high connectivity induces the usage of 4 bridges, instead of the only two of

6 Examples of M-Architecture

In this section will be presented some examples of use of the M-Architecture both in the description of the models and in the simulation of concurrent tasks.

6.1 M-Net for a Human Finger

The network (Quadfinger model) consists of 4 S-Units, 16 L-Units, and 3 M-Units. The S-Units identify the three phalanges and the metacarpus. For each of them we defined a reference frame, centered in the proximal joint and oriented in such a way that X points to the subsequent joint and Y is directed at the rotation axis.

Each one of the three joints is modeled by means of three L-Units: a pair of collateral ligaments, with high stiffness and zero rest-length, that define the rotation axis, and a volar ligament, with smaller stiffness and non-zero rest length, which avoids the occurrence of hyper-extension. The rat-tail ligament models with a soft spring the retinacular oblique ligament which cross-connects the two inter-phalangeal joints. The three M-Units represent the flexor profundus, the extensor digitorum communis, and an intersseus. The first two muscles span all the three joints, following a palmar or dorsal
pathway, respectively. The interosseus is a double-joint muscle and has an opposite effect on the two articulations (flexion of the metacarpophalangeal joint and extension of the proximal interphalangeal joint). A fragment of the network is described in the following:

```c
model OneFinger {
    S-units { 
        metacarpal(0,1,0,0,Z(0,0,1,0,0),M(0,0,0),CG(0,0,0,0,0),
        phalanx1(0,0,1,0,0,Z(0,0,1,0,0),M(0,0,0),CG(0,0,0,0,0),
        phalanx2(0,0,1,0,0,Z(0,0,1,0,0),M(0,0,0),CG(0,0,0,0,0),
        phalanx3(0,0,1,0,0,Z(0,0,1,0,0),M(0,0,0),CG(0,0,0,0,0),
    
    L-units { 
        [mpji:1:STIFF(35),RESTLI(0,0),LEN(0)]
          metacarpal(0,0,1,0,0)-phalanx1(0,0,0,0,0)
          metacarpal(0,0,1,0,0)-phalanx2(0,0,0,0,0)
          metaarpal(0,0,1,0,0)-phalanx3(0,0,0,0,0)
          phalanx1(0,0,1,0,0)-phalanx2(0,0,0,0,0)
          phalanx1(0,0,1,0,0)-phalanx3(0,0,0,0,0)
          phalanx2(0,0,1,0,0)-phalanx3(0,0,0,0,0)
          phalanx1(0,0,1,0,0)-phalanx2(0,0,0,0,0)
          phalanx1(0,0,1,0,0)-phalanx3(0,0,0,0,0)
          phalanx2(0,0,1,0,0)-phalanx3(0,0,0,0,0)
          phalanx1(0,0,1,0,0)-phalanx2(0,0,0,0,0)
          phalanx1(0,0,1,0,0)-phalanx3(0,0,0,0,0)
          phalanx2(0,0,1,0,0)-phalanx3(0,0,0,0,0)
    
    R-units { 
        ext:FMAX(9,65),LMAX(82,65),SPD(433),LEN(58.76,
        metacarpal(0,0,1,0,0)-phalanx1(0,0,0,0,0)
        phalanx1(0,0,1,0,0)-phalanx2(0,0,0,0,0)
        phalanx1(0,0,1,0,0)-phalanx3(0,0,0,0,0)
        phalanx2(0,0,1,0,0)-phalanx3(0,0,0,0,0)
        phalanx1(0,0,1,0,0)-phalanx2(0,0,0,0,0)
        phalanx1(0,0,1,0,0)-phalanx3(0,0,0,0,0)
        phalanx2(0,0,1,0,0)-phalanx3(0,0,0,0,0)
        phalanx1(0,0,1,0,0)-phalanx2(0,0,0,0,0)
        phalanx1(0,0,1,0,0)-phalanx3(0,0,0,0,0)
        phalanx2(0,0,1,0,0)-phalanx3(0,0,0,0,0)
    
    X, Y, Z are the unit-vectors of the coordinate axes of each S-Unit, R (cm) is the position vector of the origin, and CG is the center of gravity. For each L-Unit, STIFF (N/cm) is the stiffness, RESTLI (cm) is the rest-length, and LEN is the current length. For each M-Unit, FMAX (N) is the peak isometric force at the optimal length LMAX (cm), LEN is the actual length, and SPD (dimensions) is the spread coefficient.

The active tracts of L-Units and M-Units are identified by means of their insertion points. The three joints (MP, PIP, DIP) are represented by a pair of stiff S-Units, which identify the rotation axis, and a softer L-Unit, which limits the extension range.

6.2 A Concurrent Cooperating Task

As previously said, M-Net has been used to successfully model an robotic structures. In this example an industrial 6 d.o.f robot, (SAPR Erniost 0), has been modeled. L-Units were used to simulate joint limits for each degree of freedom and in general to prevent undesired extra rotations. M-Net proved to be able to generate complex and smooth end-effector trajectories.

Moreover, simulations of cooperative tasks, involving multiple multiple robots, have been tried: we defined two physically separated M-Net structures, representing two Earnst 0 robots, that are driven by a single motor plan; in this way, the cooperation between the robots is performed by a global relaxation mechanism.

In the following, a segment of the code written to implement such a rendezvous task between the two robots is listed. The robots have to grasp a box and to transfer it from one each to the other. The definition of the M-Net models of the robots (Earn1, Earn2) and of the grasped box (box) are not listed.

SEQUENCE { 
  /* The end-effector of the first robot reaches the box and grasp it in the speced point */
  go_near: Earn1, to HEBB box(0,0,0,2,2)
  grasp: Earn1, to TAKE box(5,0,0,2,2)
  TOGETHER { 
    /*
    */
```
The end-effectors of the two robots concurrently reach each other

```c
 recip_move1:Earn1,ee REACH Earn2,ee
 recip_move2:Earn2,ee REACH Earn1,ee
```

The end-effector of the second robot approaches the box in the best possible point for grasping

```c
 ge_near2: Earn2,ee NEAR box(0, 0, -32.)
 grasp2: Earn2,ee TAKE box(-5, 0, 2.)
```

When the grasping action is completed, the first robot releases the box

```c
 release: WHEN grasp2 Earn1,ee RELEASE box
 TOGETHER {

 Concurrently, the two end-effectors repel each other

 ```c
 exit1: Earn1,ee REPEL Earn2,ee TO 50
 exit2: Earn2,ee REPEL Earn1,ee TO 50
 ```

```

The statement WHEN the construct is a guarded command. The indicated construct is started after the completion of the task

Acknowledgement

The authors gratefully thank Prof. P. Morasso for his guidance, and their fellows C. Lagioni, D. Radulovich, and L. Tuano for their contribution to the development of this work.

References


