Optimization as Motion Selection Principle in Robot Action

Jean-Paul Laumond, Nicolas Mansard, Jean-Bernard Lasserre

Communications of ACM, 58(5), 2015

Optimization as Motion Selection Principle in Robot Action

Jean-Paul Laumond Nicolas Mansard Jean-Bernard Lasserre

Figure 1 *Stones and hammers do not move by themselves. Movement is a prerogative of living (and robot) system. Plants (and manipulator robots) moves to bring the world to them via self-centered movements. Animals (and mobile robots) navigate to explore the world. Human (and humanoid) actions are built from both type of movements.*

Abstract

Robots move to act. Actions are defined and operate in the physical space. At the same time motions originate in the robot motor control space. How to express actions in terms of motions? Optimization-based selection principles address these questions while opening many computational challenges.

1 Action versus motion

Movement is a fundamental characteristic of living systems (Figure 1). Plants and animals have to move to survive. Animals are distinguished from plants in that they have to explore the world to feed. The carnivorius plant remains at a fixed position to catch the imprudent insect. Plants have just to make use of self-centered motions. At the same time the cheetah has to go out looking for food.

Feeding is a parangon of action. Any action in the physical world requires self-centered movements, exploration movements or a combination of both. By analogy, a manipulator robot makes use of self-centered motions, a mobile robot moves to explore the world and a humanoid robot combines both types of motions.

Actions take place in the physical space. Motions originate in the motor control space. Robots -as any living system- access the physical space only indirectly through sensors and motors. Robot motion planning and control explore the relationship between physical, sensory and motor spaces, the three spaces that are the foundations of geometry [32]. How to translate actions expressed in the physical space into a motion expressed in motor coordinates? This is the fundamental robotics issue of inversion.

In life sciences, it is recognized that optimality principles in sensorimotor control explain quite well empirical observations, or at least in any case better than other principles [40]. The idea to express robot actions as motions to be optimized has been developed in robotics since the nineteen-seventies with the seminal work by Whitney [41]. It is now well developed in classical robot control [37], and also along new paradigms jointly developed in multidisciplinary approaches [35]. Motion optimization then appears to be a natural principle for action selection. However, we have seen in the companion paper [24] that optimality equations are most of the time intractable and numerical optimization is notoriously slow in practice. The article intends to make a short overview of recent progress in the area. We first show how robot motion optimization techniques should be viewed as action selection principles for redundant robots. In that perspective, we overview results and challenges stimulated by recent applications to humanoid robotics. The rest of the article is then devoted to inverse optimal control as a means to better understand natural phenomena and to translate them into engineering. The question opens highly challenging problems. In that context, methods based on recent polynomial optimization techniques appear complementary to classical machine learning approaches.

2 From task space to control space: power and limits of linearization

Translating actions in terms of motions expressed in the robot control space has received many wordings, from the operational space formulation [19] to the task function approach [34], to cite a few. The notion of task encompasses the notion of action expressed in the physical space. The task space may be the physical space (like for putting a manipulator end effector to some position defined in a world frame) or a sensory space (like for tracking an object in a robot camera frame). The role of the so-called task function is to make the link between the task space and the control space.

Due to the underlying highly non-linear transformations, the inversion problem is very costly to solve (minutes or hours of computation for seconds of motion). To meet the time constraints imposed by the control frequency of the robots, the problem is addressed only locally by considering the tangent spaces of both the task space and the configuration space. Such a linearization involves the Jacobian matrix [31] and resorts to all the machinery of linear algebra. The linearization is particularly interesting as the tangent space of the configuration space gathers the configuration velocities that usually contain the robot control inputs. Dynamic extensions of this principle allow considering torque-based controls [19].

The Jacobian matrix varies with the robot configuration, making the search for a trajectory non linear. However, for a given configuration, it defines a linear problem linking the unknown system velocity to the velocity in the task space given as references. From a numerical point of view, this problem is linear and can easily be solved at each instant to obtain the system velocity. The integration of this velocity from the initial configuration over a time interval draws a trajectory tending to fulfill the task. The velocity can similarly be applied in real-time by the robot to control it toward the goal. The linear problem is re-initialized at each new configuration updated with the sensor measurements and the process is iterated. This iterative principle corresponds to the iterative descent algorithms (like the gradient descent or the Newton-Raphson descent), which are used to numerically compute the zero value of a given function. However the method gives more: the sequence of descent iterations, assuming small descent steps, is a discretization of the real trajectory from the initial trajectory to the goal. The drawback of the instantaneous linearization is that it provides no look-ahead capabilities to the control, which might lead the robot to a local minimum, typically when approaching non-convex obstacles. This is the well known curse of linearization.

3 Motion selection: a question of dimensionality

The dimension of the task space can be equal, greater or lower than the dimension of the control space. For the sake of simplification, let us consider the task space as a manifold that expresses the position of the end-effector of a fully actuated manipulator. When the dimensions of both the task space and the configuration space are equal, each point in the task space defines a single configuration[1] and the task function can be used to drive the robot to a unique configuration. There is no problem of motion selection. The Jacobian matrix is square invertible and solving the linear problem is easy. The task function approach was initially proposed in this context to define admissibility properties to connect two points of the configuration space while avoiding singularities [34].

Optimization is used as motion selection principle in the other cases. The choice of the optimization criterion determines the way to invert the Jacobian matrix, as explained in the following paragraphs.

When the task space has a larger dimension than the configuration space, it is not always possible to find a configuration satisfying the task target: the task function is not onto, i.e. the Jacobian matrix has more rows than columns. It is then not possible to find a velocity in the configuration tangent space that corresponds to the velocity in the task tangent space. For instance, this is the case in visual servoing when many points should be tracked in a camera frame [4]. This is also the case in simultaneous localization and mapping when optimizing the positions of the camera with respect to the landmarks [13]. Optimization is used to find a velocity that minimizes the error in the task tangent space. The problem is then to minimize the distance to the reference task vector. Generally, the reference task vector cannot be reached. In the special case, when the reference belongs to the image space of the Jacobian, the residual of the optimization is null. This is the case in visual servoing, when the target image has been acquired from a real scene with no noise.

On the contrary, if the dimension of the task space is smaller than the dimension of the configuration space, several configurations correspond to a single task. The task function is not one-to-one, i.e. the Jacobian matrix has more columns than rows. For instance this is the case of a thirty-joint humanoid robot picking a ball with its hand: the dimension of the task space is three while the dimension of the configuration space is thirty. Several motions may fulfill the task. The system is said to be redundant with respect to the task. Optimization is then used as a criterion to select one motion among all the admissible ones. In that case, several vectors in the configuration tangent space produce the same effect in the task space. Equivalently, some velocities produce

[1] The non-linearities in the task function can generate a discrete set of configurations accomplishing the task, corresponding to several optima. In such cases, the discussion holds, but only locally.

no effect in the task space. This subset of the configuration tangent space is the kernel of the Jacobian matrix and is called the null space of the task. Any velocity in the null space leaves the task unchanged. Adding up a given configuration tangent vector satisfying the task with the null space gives the vector space of all velocities satisfying the task. The minimization problem consists in selecting one sample in this space, according to some criteria, e.g. the least-norm velocity.

In general the task function may neither be onto nor one-to-one, i.e. the Jacobian matrix is neither full row rank (i.e. its rows are not linearly independent) nor full column rank (i.e. its columns are not linearly independant). In general, no vector in the configuration tangent space satisfies the task (since the transformation is not onto), and there is an infinity of vectors that minimize the distance to the task vector in the task tangent space (since the transformation is not one-to-one). Therefore the selection problem becomes a double minimization problem: we simultaneously minimize the distance to the task and the norm of the configuration velocity. A solution for this double minimization problem is given by the Moore-Penrose pseudo-inverse [2], also called the least-square inverse. Notice that other minimization criteria may be considered in the same framework by changing the metrics in the tangent spaces. For instance, we can use weighted pseudo-inverses in which the components of the system input (columns of the Jacobian matrix) and the task residual (rows of the Jacobian) do not receive the same weight. As before the sum of the optimal vector with the null space gives the set of all solutions that are optimal for the first problem (smallest distance to the task) but only suboptimal for the second one (smallest velocity norm).

4 Optimization as selection principle

4.1 Stack of tasks for redundant systems

When a robot is redundant with respect to a task, it is interesting to allocate it a secondary task. This is the case for humanoid robots that can perform two tasks simultaneously. Consider for instance two distinct task functions dealing with the positions of the right and left hands respectively. How to check if both tasks are compatible? A simple idea consists is ordering the two tasks. At each time step of the integration process, a vector of the configuration tangent space associated to the first task is selected. Then the secondary task is considered only within the restricted velocity set lying in the kernel of the first task. The reasoning that applies to the first task also applies to the projected secondary task: the task function may be onto, one-to-one or neither of it. In particular, if it is not onto, the task is said to be singular (in the sense that the Jacobian is rank deficient). Two cases can be distinguished. If the (not projected) secondary task function is not onto, then the singularity is said to be kinematic: it is intrinsically due to the secondary task. On the opposite, if the (not projected) secondary task function is onto, then the singularity is said to be algorithmic: because of a conflict with the main task, the secondary one becomes singular [5]. Outside of algorithmic singularities, the two tasks are said to be compatible and the order between them is irrelevant.

The projection process can be iterated for other tasks, resulting in a so-called *stack of tasks* [26]. Doing so, the dimension of the successive null spaces is decreasing. The process stops either when all tasks have been processed, or as soon as the dimension of the null-space vanishes (see Figure 3). In the latter, we can still select the minimun-norm vector among those in the remaining null space.

The null space was first used in the frame of numerical analysis for guiding the descent of sequential optimization [33]. It was used in robotics to perform a positioning task with a redundant robot while taking care of the joint limits [25]. A generalization to any number of tasks was proposed in [30], and its recursive expression was proposed in [38] (see also [1] in the context of computer animation).

Here, we limit the presentation to inverse kinematics, i.e. computing the robot velocities from reference velocity task constraints. The same approach can be used in inverse dynamics, to compute the system torques [19] (typically, joint torques, but also tendon forces or other actuation parameters) from homogeneous operational constraints (typically, reference forces or accelerations). In that case, the Euclidean norm is irrelevant, and weighted inverses are generally preferred to enforce minimum energy along the motion.

4.2 Stack of tasks, quadratic programming and inequality constraints

A stack of tasks can be compared to quadratic programming: a quadratic program is an optimization problem that involves a set of linear constraints and a quadratic cost (e.g. a linear function to be approximated in the least-square sense). It is then similar to a stack with two tasks: the first task, with higher priority, would be the constraint, the secondary task would be the cost to minimize. However the similarity is not total: the constraint in a quadratic program is supposed to be admissible (at least one feasible solution exists), while it is not the case for the main task. Also a stack of tasks can be extended to more than two tasks.

Up to now, we have considered that a task corresponds to an equality in the configuration tangent space, to be satisfied at best in the least-square sense. Consider now a region defined by a set of inequalities: the robot can move freely inside the region but should stay inside it; when the task becomes infeasible, it should minimize its distance to the region in the least-square sense. Such inequality constraints can not be solved directly with the method described above.

Historically, the first solution has been to set a zero velocity in the task space when the inequality constraint is satisfied. This is the artificial potential field approach proposed by Khatib [18]: the target region is described with a low or null cost, while the cost increases when approaching to the limit of the region, following the behavior of the barrier functions used in the interior-point numerical algorithms. The gradient of the function then acts as a virtual force that pushes the robot inside the region when approaching the region boundaries while it has zero or very little influence inside the region.

For robot control, penalty functions are generally preferred to barrier functions, to prevent bad numerical behavior when the robot is pushed to the limits. For a single task or when the inequality task has the lowest priority, the obtained behavior is then always satisfactory: the robot does not have to move when the inequality is satisfied. However, it is difficult to enforce a hierarchy using this approach. The gradient-projection method [27] can be used if the inequality task has a secondary importance, in particular when enforcing the robot constraints in a very redundant context (for instance, a three dimensional reaching task performed by a 6-joint robot arm). When the inequality task has the priority, the saturation of one boundary of the task region will correspond to the allocation of one degree of freedom[2], which is allocated to fix the velocity orthogonal to the boundary. This degree of freedom is thus not available anymore for any secondary task. On the opposite, when freely moving inside the region (far from the boundaries), this degree of freedom can be used by the secondary tasks. In order to take advantage of the redundancy offered inside the region defined by the inequality constraints, the corresponding degrees of freedom should be dynamically allocated. If the inequality constraint is satisfied, the degree of freedom is left unallocated and can be used by a secondary task: the constraint is said to be inactive. If the constraint is violated, then the corresponding degree of freedom is used to satisfy the constraint at best: the constraint is said to be active.

The set of all active constraints is called the active set. Active-set-search algorithms are iterative resolution schemes searching over all possible active constraints. At each iteration, a candidate solution is computed that fits the active constraints. Depending on the status of the active and inactive constraints with respect to the candidate solution, the active set is modified and the process is iterated. Active-set-search algorithms are classical to solve inequality-constrained quadratic programs. The priority order between multiple conflicting objectives can be introduced, leading to hierarchical quadratic programs [9]

Let us illustrate the stack-of-tasks framework from the worked out example of HRP2 humanoid robot performing two simultaneous reaching tasks, while respecting equilibrium constraints. All elementary tasks are embedded into a single global trajectory. We will see that the hierarchy introduced in quadratic programming induces a structure in the task vector space. Doing so, the global trajectory appears as a composition of elementary movements, each of them characterizing a given task (or subtask). Reverse engineering can then be used to identify the "meaning" of the motion, i.e. the various tasks the motion is embedding.

[2]A degree of freedom is a linear combination of controls in the configuration tangent space.

5 Motion as action signature: a worked out example in humanoid robotics

We overview here two practical applications of the stack of tasks on the humanoid robot HRP2[3]. The first one shows how to express complex actions while involving all body segments and respecting physical constraints. The second application shows how it is possible to recognize actions from motion observation using reverse engineering techniques.

5.1 From action to motion: the optimization based selection principle at work

The stack of tasks is a generic tool to generate and to control a motion of the robot. Given an initial configuration, a motion is generated by adding a set of tasks into the stack and integrating the resulting velocity until the convergence of all the active tasks. The stack of tasks can be used in various robotics scenarios. In humanoid robotics, classical tasks deal for instance with reaching (expressed as the placement of an end-effector), visual servoing (expressed as the regulation of the gaze on the position of an object in the image plane of the robot camera), or quasi-static balance (expressed as the regulation of the center-of-mass in such a way that its projection on the floor lies inside the support polygon of the feet).

For example, the motion in Figure 2-(top) is generated by constraining the two feet and the center of mass to remain to their initial positions and by setting two tasks to control the right hand and the gaze both to the ball in front of the robot. The robot bends forward to reach the ball. Doing so, it pushes the center of mass forward. The left hand moves backward to compensate for this motion of the center of mass. The motion of the left hand does not answer a specific action. It is a side-effect of the balance maintenance.

The motion can easily be modified by setting new tasks or changing the desired value of the active tasks. For example, the motion in Figure 2-(bottom) is generated by adding a task that regulates the position and orientation of the left hand to the final placement of the left hand in the first scenario. This new task is a reaching task: the left hand has to reach a goal. The two

[3]A detailed presentation appeared in [12].

movements of the left hand in both scenarios look very similar, but their meanings are different. In the first case, the motion is not intentional: the left hand moves to regulate the center-of-mass position; its motion is then a side-effect of the other tasks. In the second case, the motion is intentional: the left-hand explicitly moves to reach a given target. A careful analysis of slight differences between the two left hand motions allows to eliminate the ambiguity. This is made possible by a reverse engineering approach.

5.2 From motion to action: a reverse-engineering approach of action recognition

The hierarchy artificially decouples the tasks of the stack in order to prevent any conflict between two different tasks. A side effect is that the trajectory into a given active task space is not influenced by any other task. For example, on Figure 2-(bottom) the stack of tasks enforces a decoupling for the left hand, which moves independently of the two other tasks. The trajectory in one task space then constitutes a signature of the activity of the task in the generation of the robot motion.

Consider now the following problem: we observed the motion of a system whose possible controllers are known. Observing only the joint trajectory, the question is to reconstruct which of the possible controllers were active and which were the associated parameters. Recovering one task is easy: the configuration trajectory is projected in all the candidate task spaces using the corresponding task function. The best task is selected by fitting the projected trajectory with the task model (once more, the fitting and thus the selection is done by optimization).

However, if the stack of tasks artificially decouples the active tasks, some coupling between the candidate tasks may occur: for example, there are a lot of similarities between the trajectories of the wrist and the elbow due to their proximity in the kinematic chain. These similarities can lead to false positives in the detection. To avoid this problem, only the most relevant task is chosen first. The motion due to this task is then canceled by projecting the configuration trajectory in the null space of the detected task. The detection algorithm then iterates until all the tasks have been found, i.e. until the remaining quantity of movement after successive projections is null [12].

This detection algorithm can be used for example to disambiguate the two similar-looking motions performed in Figure 2, without using any contextual information. An illustration of the successive projections is given in Figure 3. The tasks are removed in the order given by the detection algorithm. The right-hand task is removed first (second row), followed by the center of mass (third row): this cancels most of the motion of the left hand because the coupling between the three tasks is important; however a small part of left-hand movement remains. On the contrary, the head movement, which is nearly decoupled from the right-hand and center-of-mass, remains important. It is totally nullified after removing the gaze task (fourth row). The remaining motion of the left hand can only be explained by the left-hand task, which is detected active and then removed. Finally, the two feet constraints are detected and removed. The effect of the first foot removal (sixth row) is noticeable.

The algorithm achieves very good performances to recognize actions and to tell the differences between similar-looking robot motions. Beyond robotics, the method can be applied to human action recognition. However, it requires a critical prerequisite: the knowledge of the optimality principles grounding the motion generation of intentional actions. Indeed, the algorithm is based on action signatures that are the typical results of a particular cost function. The approach also requires a computational model of the coordination strategies used by the human to compose several simultaneous motion primitives. They are the promising routes for future researches combining computational neuroscience and robotics.

At this stage, we have seen how optimization principles and the notion of tasks help to ground a symbolic representation of actions from motions: an action is viewed as the result of an optimization process whose cost represents the signature of the action. The next section addresses the dual problem of identifying action signatures from motions.

6 Inverse Optimal Control

Let us introduce the section by a case study taken from humanoid robotics. Suppose we want a humanoid robot to walk as a human, that is, following human-like trajectories. So the question is: what are the computational foundations of human locomotion trajectories? In a first stage, we showed that locomotor trajectories are highly stereotypical across repetitions and subjects. The methodology is based on statistical analysis of a huge motion capture data basis of trajectories (7 subjects, more that 1500 trajectories) [15]. Next, in a second stage, it is assumed that human locomotion trajectories obey some optimality principle. This is a frequent hypothesis in human or animal motion studies. So the question is: which cost functional is minimized in human locomotion? In practice we consider that the human and the robot obey the same model, i.e. we know precisely the differential equation that describes the motions under some control action, and the constraints that the state of the system should satisfy. The data basis of trajectories is available. Then based on this knowledge, determining a cost functional that is minimized in human locomotion becomes an inverse optimal control problem.

Pioneering work for inverse optimization in control dates back to the nineteen-sixties in systems theory applied to economics [29]. Similarly, for optimal stabilization problems, it was known that every value function of an optimal stabilization problem is also a Lya-

Right-hand reaching while enforcing the balance: the left hand had moved only to correct the balance.

Simultaneous right-hand and left-hand reaching: the target imposed to the left hand is the final position reached in the previous (right-hand only) movement.

Figure 2: *Examples of motions generated by the stack of task. Top: The stack of tasks is composed of three constraints (both feet and center of mass should remain at a fixed position; all of them are always feasible and satisfied) and two tasks to control the gaze and the right hand. At the final configuration, the robot has reached the ball with its right hand and the ball is centered in the robot field of view. The left hand had moved only to regulate the position of the center of mass. Bottom: The motion is similar but a task has been added to control the position of the left hand: the desired position imposed to the left hand is the final position reached by the left hand in the previous movement. The two motions look very similar, but their "meanings" are different. In the top motion, the left hand moves to regulate the balance; in the bottom motion, the left hand moves to reach a specific position.*

punov function for the closed-loop system. Freeman and Kokotovic [10] have shown that the reciprocal is true: namely, every Lyapunov function for every stable closed-loop system is also a value function for a meaningful optimal stabilization problem.

In static optimization, the direct problem consists in finding in some set K of admissible solutions a feasible point x that minimizes some given cost function f. We state the associated inverse optimization problem as follows: given a feasible point y in K , find a cost criterion g that minimizes the norm of the error $(g-f)$, with g being such that y is an optimal solution of the direct optimization problem with cost criterion g (instead of f). When f is the null function, this is the static version of the inverse optimal control problem. Pioneering works date back to the nineteen-nineties for linear programs, and for the Manhattan norm. For the latter, the inverse problem is again a linear program of the same form. Similar results also hold for inverse linear programs with the infinite norm. In Heuberger [14], the interested reader will find a nice survey on inverse optimization for linear programming and combinatorial optimization problems. Note that, in inverse optimization, the main difficulty lies in having a tractable characterization of global optimality for a given point and some candidate cost criterion. This is why most of all the above works address linear programs or combinatorial optimization problems for which some characterization of global optimality is available and can sometimes be effectively used for practical computation. This explains why inverse (non linear) optimization has not attracted much attention in the past.

Recently, some progress has been made in inverse polynomial optimization, that is, inverse optimization problems with polynomial objective function and semi-algebraic set as feasible set of solutions [22]. Powerful representation results in real algebraic geometry [21] permit to describe the global optimality constraint via some certificate of positivity. These can be stated as linear matrix inequalities (LMIs) on the unknown vector of coefficients of the polynomial cost function. The latter set is a convex set on which we can optimize efficiently via semidefinite programming [23], a powerful technique of convex optimization. Then, we can show that computing an inverse optimal solution reduces to solving a hierarchy of semidefinite programs of increasing size.

Back to the inverse optimal control problem for anthropomorphic locomotion, we can consider a basis of functions to express the cost function candidate. The method proposed in [28] is based on two main algorithms: an efficient direct multiple shooting technique to handle optimal control problems, and a state of the art optimization technique to guarantee a match between a solution of the (direct) optimal control problem and measurements. Once an optimal cost function has been identified (in the given class of basis functions), we can implement a direct optimal control solution on the humanoid robot. So far, the method is rather efficient at least on a sample of test problems. However it requires to define an a priori class of basis functions. Moreover, at each iteration of the algorithm, the direct shooting method provides only a local optimal solution. Thus, there is no guarantee of global optimality.

An alternative way to consider the problem is to extend the methodology developed for inverse polynomial

Figure 3: *The original movement refers to Figure 7 (bottom): successive projection of the motion after detecting each of the seven tasks. From top to bottom: original movement; removing the right-hand task; removing the center-of-mass task; removing the gaze task; removing the left-hand task; removing the left foot task; removing the right-foot task. On the last row, all the tasks are canceled, the projected movement is totally nullified.*

optimization [22] to the context of inverse optimal control. Note that the Hamilton-Jacobi-Bellman (HJB) equation is the perfect tool to certify global optimality of a given state-control trajectory whenever the optimal value function is known. So the basic idea is to use a relaxed version of the HJB-optimality equation as a certificate of global optimality for the experimental trajectories stored in the data base. Then, the optimal value function, which is generally assumed to be continuous, can be approximated on a compact domain by a polynomial. If we search for an integral cost functional whose integrand h is also a polynomial, then this certificate of global optimality can be used to compute h and an associated (polynomial) optimal value function by solving a semidefinite program. Proceeding as in [22], we solve a hierarchy of semidefinite programs of increasing size. At each step of this hierarchy, either the semidefinite program has no solution or any optimal solution h is such that the trajectories of the data base are global optimal solutions for the problem with polynomial cost function h as integrand. The higher in the hierarchy the better is the quality of the solution (but also at a higher computational cost)

Apart from polynomial optimization techniques, other approaches have been recently introduced with a geometric perspective of optimal control theory. Significant results have been obtained in the context of pointing motions [3]: based on Thom transversability theory, the cost structure is deduced from qualitative properties highlighted by the experimental data. These can be for instance the characterization of inactivity intervals of the muscles during the motion. Such a qualitative approach has been also successfully applied to human locomotion [6].

We have introduced the inverse optimal control problem from the perspective of biomimetic approaches to robot control. The question is: how to synthesize natural motion laws to deduce from them optimal control models for robots? We emphasized recent developments in inverse polynomial optimization. It should be noticed that this viewpoint is far from covering all the approaches to inverse optimal control. Inverse optimal control is also an active research area in machine learning. In the context of reinforcement learning [16, 20], inverse reinforcement learning constitutes another resolution paradigm based on Markov decision processes with spectacular results on challenging problems such as helicopter control [7]. The method corpus comes from stochastic control (see [20] and references therein.)

7 Computation: a practical or theoretical problem?

In computer animation optimization-based motion generation is experienced as giving excellent results in terms of realism in mimicking nature. For instance, it is possible to use numerical optimization to simulate very realistic walking, stepping or running motions for human-like artifacts. These complex body structures include up to 12 body segments and 25 degrees of freedom [36]. At first glance, the approach a priori applies to humanoid robotics. An example where the robot HRP2 steps over a very large obstacle is given in Figure 4-(top).

However robotics imposes physical constraints that are absent from the virtual worlds and that require computation performance. Biped walking is a typical example where the technological limitation implies a search of alternative formulations. The bottleneck is the capacity of the control algorithm to meet the real-time constraints.

In the current model-based simulation experiments the time of computation is evaluated in minutes. Minute is not a time scale compatible with real-time. For instance, computation time upper-bounds of few milliseconds are required to ensure the stability of a standing-up humanoid robot. So taking advantage of general optimization techniques for real-time control necessary requires to build simplified models or to develop dedicated methods. The issue constitutes an active line of research combining robot control and numerical optimization.

An example is given by the research on walking motion generation for humanoid robots. The most popular walking pattern generator is based on a simplified model of the anthropomorphic body: the linearized inverted pendulum model. It was introduced in [17] and developed for the HRP2 humanoid robot. The method is based on two major assumptions: (1) the first one simplifies the control model by imposing a constant altitude of the center of mass; (2) the second one assumes the knowledge of the foot prints. Assumption (1) has the advantage to transform the original nonlinear problem into a linear one. The corresponding model is low dimensioned and it is possible to address (1) via an optimization formulation [8]. With this formulation, assumption (2) is no longer required. The method then gives rise to an on-line walking motion generator with automatic foot step placement. This is made possible by a linear model-predictive control whose associated quadratic program allows much faster control loops than the original ones in [17]. Indeed, running the full quadratic program takes less than 1ms with state of the art solvers. More than that, in this specific context, it is possible to devise an optimized algorithm that reduces by 100 the computation time of a solution [8].

An example of this approach is given in Figure 4-(bottom) that makes the real HRP2 step over an obstacle. The approach based on model reduction enables the robot to be controlled in real-time. However, the reduced model does not make a complete use of the robot dynamics. The generated movement is less optimal than when optimizing the robot whole-body trajectory. Consequently, it is not possible to reach the same performances (in this case, the same obstacle height): the whole-body optimization enables the robot to reach higher performances, but only offline. Reaching the same performance online requires either more powerfull computers (running the same algorithms) or more

clever algorithms.

8 Conclusion

The notion of robot motion optimality is diverse in both its definitions and its application domains. One goal of this overview was to summarize several points of view and references spread out over various domains: robotics, control, differential geometry, numerical optimization, machine learning, and even neurophysiology.

The objective was to stress the expressive power of optimal motion in robot action modeling and to present current challenges in numerical optimization for real-time control of complex robots, like the humanoids. A second objective was to report recent issues in inverse optimal control. While its stochastic formulation is popular in machine learning, other paradigms are currently emerging in differential geometric control theory and polynomial optimization.

As testified in a companion paper [24], robotics offers rich benchmarks for optimal control theory. Due to real-time computation constraints imposed by effective applications, robotics also induces challenges to numerical optimization. The difficulty for roboticists is to find the right compromise between generality and specificity. General algorithms suffer from the classical curse of dimensionality that constitutes a bottleneck for robot control. Therefore, they may be used for off-line motion generation, but they are inefficient for real-time applications. Real-time robot control requires very fast computations. It requires dedicated numerical optimization methods. We have seen how bipedal walking illustrates this tension between generality and specificity. Roboticists are today asking optimization theorists for more efficient algorithms, while they are developing at the same time a specific know-how to this end.

Last but not least, let us conclude by referring to a controversy introduced by neurophysiologist K. Friston. In a recent paper [11], he asks the provocative question: "Is optimal control theory useful for understanding motor behavior or is it a misdirection?" He opposes to optimal control the competitive notion of active inference. While the paper is mainly dedicated to motor control in life sciences, the issue is of crucial and utmost interest for roboticists and pleads to a reinforcement of the cooperation between life and engineering sciences.

Acknowledgments

The paper benefits from comments by Quang Cuong Pham, from a carefull reading by Joel Chavas, and above all, from the quality of the reviews. The work has been partly supported by ERC Grant 340050 Actanthrope, by a grant of the *Gaspar Monge Program for Optimization and Operations Research* of the *Fdération Mathématique Jacques Hadamard* (FMJH) and by the grant ANR 13-CORD-002-01 Entracte.

References

[1] P. Baerlocher and R. Boulic. An inverse kinematic architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, 6(20):402–417, 2004.

[2] A. Ben-Israel and T. Greville. *Generalized inverses: theory and applications*. CMS Books in Mathematics. Springer, 2nd edition, 2003.

[3] B. Berret, C. Darlot, F. Jean, T. Pozzo, C. Papaxanthis, and J.-P. Gauthier. The inactivation principle: Mathematical solutions minimizing the absolute work and biological implications for the planning of arm movements. *PLoS Comput Biol*, 4(10):e1000194, 2008.

[4] F. Chaumette and S. Hutchinson. Visual servo control, Part I: Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4):82–90, 2006.

[5] S. Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Trans. on Robotics and Automation*, 13(3):398–410, 1997.

[6] Y. Chitour, F. Jean, and P. Mason. Optimal control models of goal-oriented human locomotion. *SIAM Journal on Control and Optimization*, 50:147–170, 2012.

[7] A. Coates, P. Abbeel, and A. Ng. Apprenticeship learning for helicopter control. *Communications of the ACM*, 52(7):97–105, 2009.

[8] D. Dimitrov, P.-B. Wieber, O. Stasse, H. Ferreau, and H. Diedam. An optimized linear model predictive control solver. In *Recent Advances in Optimization and its Applications in Engineering*, pages 309–318. Springer, 2010.

[9] A. Escande, N. Mansard, and P.-B. Wieber. Hierarchical quadratic programming. *The International Journal of Robotics Research*, 33(7):1006–1028, 2014.

[10] R. Freeman and P. Kokotovic. Inverse optimality in robust stabilization. *SIAM J. Control Optim*, 34:1365–1391, 1996.

[11] K. Friston. What is optimal about motor control? *Neuron*, 72(3):488–498, 2011.

[12] S. Hak, N. Mansard, O. Stasse, and J.-P. Laumond. Reverse control for humanoid robot task recognition. *IEEE Trans. Sys. Man Cybernetics*, 42(6):1524–1537, 2012.

[13] C. Harris and J. Pike. 3d positional integration from image sequences. *Image and Vision Computing*, 6(2):87–90, 1988.

[14] C. Heuberger. Inverse combinatorial optimization: A survey on problems, methods and results. *Journal of Comb. Optim*, 8:329–361, 2004.

[15] H. Hicheur, Q. Pham, G. Arechavaleta, J.-P. Laumond, and A. Berthoz. The formation of trajectories during goal-oriented locomotion in humans. Part I: A stereotyped behaviour. *European Journal of Neuroscience*, 26(8):2376–2390, 2007.

[16] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[17] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *IEEE Int. Conf. on Robotics and Automation (ICRA'03)*, pages 1620–1626, 2003.

[18] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[19] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *The International Journal of Robotics Research*, 3(1):43–53, 1987.

[20] J. Kober, J. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11), September 2013.

[21] J.-B. Lasserre. *Moments, Positive Polynomials and Their Applications*. Imperial College Press, London, 2010.

[22] J.-B. Lasserre. Inverse polynomial optimization. *Math. Oper. Res.*, 38(3):418–436, August 2013.

[23] J.-B. Lasserre and M. Anjos, editors. *Semidefinite, Conic and Polynomial Optimization*. Springer, 2011.

[24] J. Laumond, N. Mansard, and J. Lasserre. Optimality in robot motion (1): optimality versus optimized motion. *Communications of the ACM*, September 2014.

[25] A. Liégeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Transactions on Systems, Man and Cybernetics*, 7:868–871, 1977.

[26] N. Mansard and F. Chaumette. Task sequencing for sensor-based control. *IEEE Trans. on Robotics*, 23(1):60–72, 2007.

[27] E. Marchand and G. Hager. Dynamic sensor planning in visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'98)*, pages 1988–1993, 1998.

[28] K. Mombaur, A. Truong, and J.-P. Laumond. From human to humanoid locomotion: an inverse optimal control approach. *Autonomous Robots*, 28(3):2010, 2010.

[29] K. Mordecai. On the inverse optimal problem. mathematical systems theory and economics, i, ii. *Lecture Notes in Oper. Res. and Math. Econom.*, 11, 12, 1969.

[30] Y. Nakamura, H. Hanafusa, and T. Yoshikawa. Task-priority based redundancy control of robot manipulators. *The International Journal of Robotics Research*, 6(2):3–15, 1987.

[31] R. Paul. *Robot Manipulators: Mathematics, Programming, and Control*. MIT Press, Cambridge, MA, USA, 1st edition, 1982.

[32] H. Poincaré. On the foundations of geometry (1898). In W. Ewald, editor, *From Kant to Hilbert: A Source Book in the Foundations of Mathematics*. Oxford Univ. Press, 1996.

[33] J. Rosen. The gradient projection method for nonlinear programmimg, part ii, nonlinear constraints. *SIAM Journal of Applied Mathematics*, 9(4):514–532, 1961.

[34] C. Samson, M. L. Borgne, and B. Espiau. *Robot control: the task function approach*. Clarendon Press, 1991.

[35] S. Schaal, P. Mohajerian, and A. Ijspeert. Dynamics system vs optimal control: a unifying view. *Prog. Brain Research*, 165:425–445, 2007.

[36] G. Schultz and K. Mombaur. Modeling and optimal control of human-like running. *Mechatronics, IEEE/ASME Transactions on*, 15(5):783–792, 2010.

[37] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer, 2009.

[38] B. Siciliano and J.-J. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *IEEE Int. Conf. on Advanced Robot*, 1991.

[39] O. Stasse, B. Verrelst, B. Vanderborght, and K. Yokoi. Strategies for humanoid robots to dynamically walk over large obstacles. *IEEE Transactions on Robotics*, 25:960–967, 2009.

[40] E. Todorov. Optimality principles in sensorimotor control. *Nature Neuroscience*, 7(9):905–915, 2004.

[41] D. Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems*, 10(2):47–53, 1969.

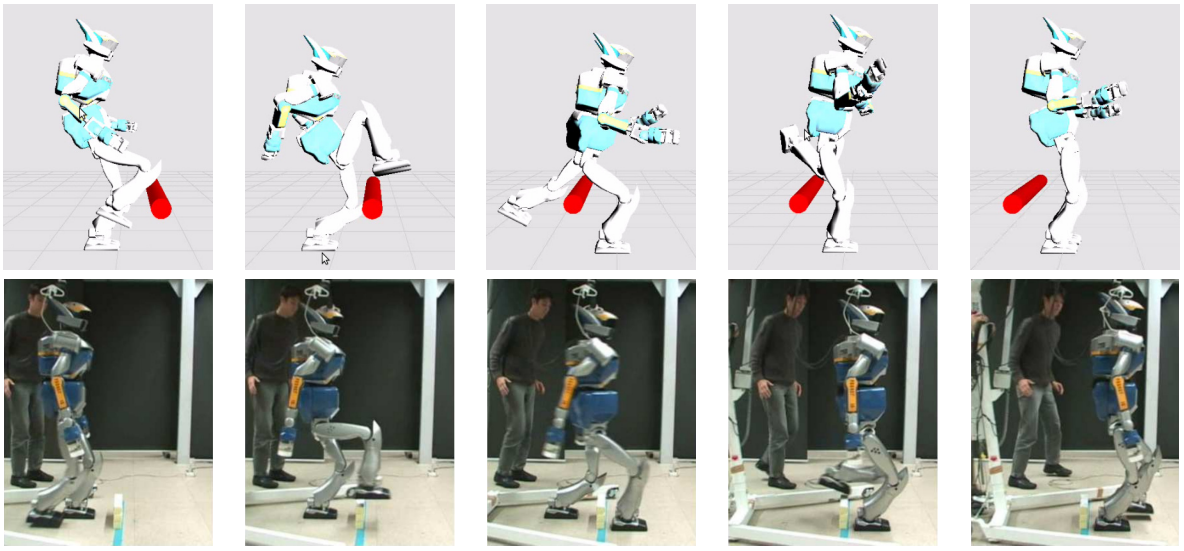

Figure 4: *Two stepping movements obtained with (top) a whole-body trajectory optimization [36] (courtesy from K. Mombaur) and (bottom) a linearized-inverted-pendulum based walking pattern generator [17] (courtesy from O. Stasse [39]). The whole-body optimization enables the robot to reach higher performances but the numerical resolution is yet too slow to obtain an effective controller.*