Pareto Optimal Control for Natural and Supernatural Motions

Shailen Agrawal Michiel van de Panne
University of British Columbia

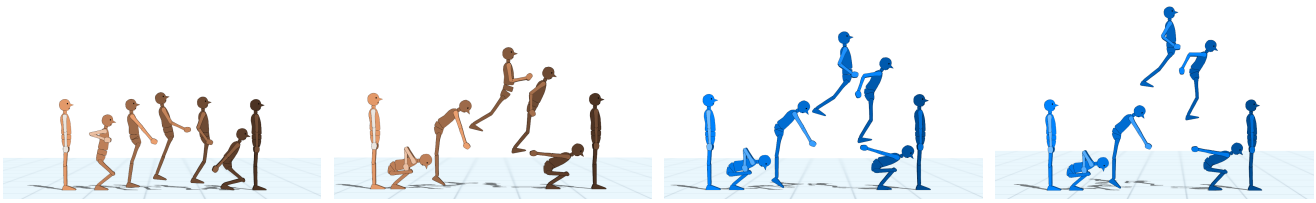


Figure 1: *Pareto-Optimal controllers with respect to jump height and control effort. The first two images on the left represent natural motions while the two images on the right represent super-natural motions.*

Abstract

Optimization is a natural tool for designing natural motion control strategies. However, optimal motions can be expensive to compute. Furthermore, we are often interested in knowing an entire family of optimal motions rather than single motion. For a motion such as a jump, the solution family of interest is described by the pareto-optimal front that defines the trade-off between effort and jump height. In this paper we explore algorithms for computing a set of controllers that span the pareto-optimal front for jumping motions. Once computed, these controllers can then drive physics-based simulations in real time. We also develop supernatural jump controllers through the optimized introduction of external forces. We show that the pareto-optimal front can naturally span both natural and supernatural regimes. This allows for controllers that can naturally transition from physics-based motions to motions assisted by external forces as the task demands increase.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: human simulation, control, pareto front optimization

1 Introduction

Humans are generally efficient at achieving their motion goals. This makes optimization a natural tool for the design of physics-based motions. While the focus of much work in this area has been to produce individual optimized motion instances, most motions have goals that are naturally parameterized, such as walking speed or jump height. In this paper we investigate several possible algorithms for computing a set of solutions that span a pareto-optimal front. Pareto optimality, for our problem domain, is a state of high dimensional value assignment to samples such that it is impossible to make an improvement with respect to any particular dimension without making at least one of the dimensions in the sample's value assignment worse off. In the context of a human jump that trades-off effort for jump height, the goal of the algorithms is to automatically compute a set of solutions that span the range from low-effort low-height jumps to large-effort high-height jumps. By aiming to simultaneously compute a range of solutions, we can expect that there are significant efficiencies to be gained.

Pareto optimal solutions can also be used in other related ways, such as exploring the trade-offs between various terms in an objective function. Many optimization problems for animation problems have multi-term objective functions with relative weightings that are defined by hand. These can alternatively be approached using multi-objective optimization, with each term contributing a dimension towards a multi-dimensional pareto-optimal front. Our height-vs-effort optimization can also be thought of in these terms. In this paper we leave the pursuit of the higher-dimensional ($d > 2$) pareto-optimal fronts as future work, although we expect that many of the core ideas and algorithms will still apply.

In this paper we also explore the idea of motions which are assisted by external forces on an as needed basis. This force is applied during the airborne rising phase of a jump and allows for jump heights that would otherwise be unachievable. Unlike kinematic approaches for altering motions, the controller-based solution still allows the motion to evolve in accordance with other interactions with the environment. We show that the pareto-optimal front can be computed in a way that spans both the natural and supernatural regimes of motion.

2 Related Work

Many control strategies have been developed for physics-based characters over the past two decades, often with a focus on human locomotion. A recent survey can be found in [Geijtenbeek and Pronost 2012]. The controllers are often structured around a sequence of motion phases, represented as states in a finite state machines, with further continuous feedback laws added within each state, such as those that make use of foot placement for balance [Raibert and Hodgins 1991; Hodgins et al. 1995; Laszlo et al. 1996; Yin et al. 2007; Tsai et al. 2010]. Other common design elements for physics-based motion controllers include the use of reference data from motion capture [Sok et al. 2007; Yin et al. 2007; Muico et al. 2009; Lee et al. 2010; Kwon and Hodgins 2010; Ye and Liu 2010; Da Silva et al. 2008; Liu et al. 2010; Coros et al. 2011] or, alternatively, to develop objective functions that are then optimized online or offline [Liu and Popović 2002; Macchietto et al. 2009; Wang et al. 2009; de Lasa et al. 2010; Wu and Popović 2010; Borno et al. 2013].

Covariance matrix adaptation (CMA)[Hansen 2006] has become a popular generic method for the derivative-free optimization problems that commonly arise when shooting-style methods are applied to the synthesis of controllers for physics-based characters.

Notable examples of methods which use offline optimization to develop parameterized control solutions include [Coros et al. 2011; Wang et al. 2012]. The work of Wang [2012] is particularly relevant to our work as it uses CMA to solve for optimal energy motions at a discrete set of locomotion speeds. The desired speed is approximately enforced by establishing a velocity bracket around the target speed and assigning a penalty for going outside of this bracket. Taken together, the solutions effectively define a pareto-optimal front. We include a similar algorithm in our set of pareto-optimal front algorithms for use as a performance benchmark.

A variety of algorithms have been developed specifically for multi-objective optimization [Deb et al. 2002; Iorio and Li 2004; Igel et al. 2007]. MOO $(1 + \lambda)$ CMA-ES [Igel et al. 2007] is of particular interest because of its invariance properties. However, in its default form we found it to be ill suited for our multi-objective optimization problems. We adapt it in several ways in order to produce an algorithm that works for the domain of highly constrained character controllers and which can span the regimes of natural and supernatural motions.

Kinematic momentum scaling methods have been proposed to directly synthesize modified trajectories for characters that then represent extreme physical capabilities [Yamane and Sok 2010]. Our work takes the alternative approach of integrating external forces into the controller that can be used when the internal forces are insufficient to achieve a task. In this way the controllers are designed to span both natural and supernatural motion regimes, and the character can at all time still interact in a dynamic fashion with its environment.

3 Controller and Task Representation

In this section we describe the parameterization of the pareto-optimal control problem. The input consists of a successful controller instance that is used to seed the optimization. This controller was designed manually but this process can be automated by using motion capture data and a sampling based approach [Liu et al. 2010] or by using video-based methods [Vondrak et al. 2012]. The free parameters consist of the joint target angles for the hips, knees, back, abdominal region, shoulder and elbows for all phases; joint target angles for the ankles for take off and rising phase; and the external force parameters. Thus we optimize for 34 free parameters.

The simulations are performed using a sagittal plane 2D human model that is well suited to the standing jump motion that we investigate. The model has 17 links, weighs 80 kg, and is 160 cm tall. The motion is simulated using the Vortex [CMLabs] physics engine. The controller uses left-right symmetry. All joints are controlled using PD Controllers with fixed, manually-determined gains. The PD controller gains were manually tuned to create natural looking motions arising from some test controllers which performed a jump. The tuning was performed with the aim of creating a realistic take off where the character has to power the jump by throwing its arms forward and extending its knees and hips during take off, have a smooth follow through while creating a compliant landing. Jumps can become unnatural if the ankles are allowed to be too stiff, which allows a jump to be achieved with a single powerful push of the ankles. The feet, pelvis and upper torso are servoed to target angles that are expressed in the world coordinate frame while all other joints servo using local joint angles.

3.1 Controller phases

A parameterized controller is used in conjunction with a forward dynamics simulation to produce the motions. The motion is broken into different phases as shown in the Figure 2.

The jump starts with a standing phase which has a timed transition into a crouched take-off position. Target angles over time are modeled using piecewise linear trajectories. A virtual force, implemented using internal torques, is applied to the COM of the character to maintain balance similar to the virtual force application in [Coros et al. 2010]. After spending some time in the crouched phase the controller transitions to a take-off phase during which the character rapidly extends its arms upwards and straightens out its knees and ankles to produce a vertical and forward momentum (if desired). Once the character breaks contact with the ground the controller transitions into flight phase. While in air, a simple time prediction strategy determines the time taken to reach the peak point of the motion. Once the peak height of the motion gets attained the character transitions to a falling phase. During this phase, again, a simple prediction strategy determines both the approximate time and location of touch-down. Inverse kinematics is used to target the feet to their predicted touch-down location. After coming into contact with the ground the character transitions into a landing phase where a virtual force is applied to the COM of the character to bring the horizontal momentum to zero while the target angles for this phase raise the character to attain a standing posture. For more details please refer [Agrawal et al. 2013].

3.2 Task Representation

An in-place jump motion that achieves a given jump height is optimized for minimal control effort with the help of a penalty function that constrains the landing position as well as the maximum head height achieved during the jump (eq. (1)). The landing constraint term, C_L, evaluates to zero when satisfied and returns a constant and large failure penalty when violated (eq. (3)). This is equivalent to rejecting solutions that do not satisfy the landing constraint. However, we do not explicitly discard these evaluations but rather rely on the optimization algorithm to perform any required adaptations by providing feedback via the fitness assessment. The height of the jump of the character can be constrained by constraining the head position at the peak point of the jump. The head constraint function, C_H, is modeled using a penalty that evaluates to zero when the constraint is satisfied to within a desired value, while rising quadratically when it is violated as seen in eq. (2). The treatment of the jump height as an independent variable allows for a default algorithm that uses control-effort optimizations for a set of fixed target-height bins.

$$f_{\mathrm{H}}(q_0) = C_H + C_L \tag{1}$$

$$C_H = \begin{cases} w_p \times (h_p - h_o)^2 & \text{if } |h_p - h_o| > h_r \\ 0 & \text{if } |h_p - h_o| \leq h_r \end{cases} \tag{2}$$

$$C_L = \begin{cases} f_p & \text{if } |l_p - l_o| > l_r \\ 0 & \text{if } |l_p - l_o| \leq l_r \end{cases} \tag{3}$$

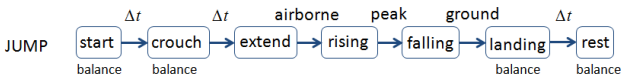


Figure 2: *Motion control phases*

In the above, h_r and l_r define the bracket of values for which constraint satisfaction holds true. h_o and l_o are the constraint target values, h_p and l_p are the height and location of landing of the current sample. w_p is the weight which scales the height constraint. The results are found to be not sensitive to this weight as long as a large enough value is used. We used a value of 5000 for w_p for all our experiments. f_p was set to a really large value of 100000 to ensure that it is always greater than any possible fitness value.

We experimented with a simple joint squared torque model for use as an effective energy metric. However, since the "effort" spent per joints is different as noted by [Wang et al. 2009], we use a weighted effort metric which is of the following form :

$$e_{\text{JE}}(q) = \sum_i w_i \times ||\tau_i||^2 \tag{4}$$

The weights w_i before normalization are described in Table 1. These values are based on experimentation and correspond to an intuitive assumption of the effort spent in a particular joint. This weighting was found to produce a more natural looking result than an unweighted effort metric for creating jumping motions. As an example, unweighted effort metric often results in jumps where the character powers its jumps using unnaturally high contribution from hips and knees as compared to its ankles. Such insights were used to come up with the weights we have used in our examples. Weights are normalized so that $\sum w_i = n$, where n is the number of joints.

hips	knees	ankles	shoulders	elbows
2.5	2	1	1	0.5
wrists	head	neck	back	abs
0.25	1	1	2.5	3

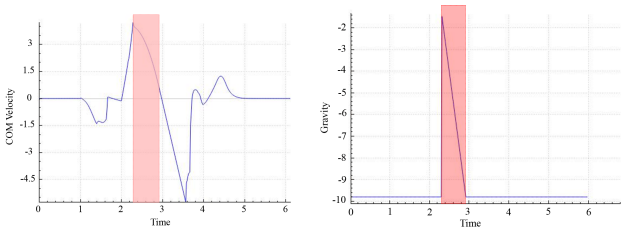
Table 1: *Weights used for computing the internal joint effort metric*

3.3 Natural and Supernatural Motion

We want to generate natural motions when possible and motions assisted by external forces only on an "as needed" basis. These supernatural motions can be used to animate super-human and imaginary characters. Super-natural motions are generated by allowing for reduced gravity. Gravity is reduced to a specified value at time of take-off. Right after that the gravity linearly increases to its default value of $-9.8m/s^2$ as can be seen from Figure 3. After this point the system maintains this default value for the rest of the jump. Thus, there are two parameters which define the extent to which an external assist is provided to the motions: the magnitude of the initial reduced gravitational force and the time duration before which gravity returns to normal. Reducing gravity in such a manner is functionally equivalent to applying an external force to the center of mass of the character. This model of application of external force by modifying gravity was used because of convenience in representation and implementation in the physics engine.

4 Optimization Framework

A parametric family of pareto-optimal controllers spanning a range of fitness values is desirable since once such a family is precomputed, an appropriate controller for the task at hand can be selected from this family in real-time. Samples on the pareto-optimal front can be selected on the basis of their suitability for a certain task. We propose an optimization procedure to pre-compute this pareto-optimal front of controllers. The proposed algorithms are applied to a working example of simulated in-place standing jumps. Additionally, we generate supernatural motions with an assistive



(a) COM Velocity Profile (b) Gravity Profile

Figure 3: *(a) A plot of COM velocity vs time is shown here. The red rectangle indicates the region where external forces are active. (b) The sudden jump in gravity occurs at take-off. The red rectangle highlights the supernatural region.*

external force when a particular jump height is unattainable by the character. For the controllers generated within this supernatural region we still want the character to minimize its use of the assistive force by maximizing the use of internal energy towards achieving the task. The goal is to produce super-natural jumps that look as realistic as possible.

In this section we describe our proposed optimization framework as well as a simple binned single objective optimization (BSOO) strategy for comparison. We first describe the BSOO scheme which classifies jumps into bins of varying jump heights and uses a single objective optimization for each bin. This type of approach has been previously used in developing controllers for character locomotion, e.g., [Wang et al. 2012]. Instead of generating a single solution from each optimization run our approach instead recovers a complete pareto-optimal front in a single optimization process. We also show that the BSOO strategy is inefficient when compared to the multiobjective algorithm which can avoid sampling redundancies that arise in the BSOO strategy. We adapt an existing multi-objective optimization technique [Igel et al. 2007] (MOO) to this effect and apply it to a high-dimensional and constrained physics-based character animation problem.

4.1 Binned Single Objective Optimization (BSOO)

The simplest way to explore the pareto-optimal front of the controllers representing the trade-offs between two different objectives is to categorize one of them into bins. A single objective optimization for the second objective can then be performed for the task specified by the first objective bin. For our example we split the task (jump height) into 16 uniformly spaced bins, each representing a height constraint. All bins have the same landing constraint.The objective function f_H defined in eq.(1) is combined with an effort metric for joint energy (eq.(4)) and a supernatural effort metric to produce the following :

$$f_{\text{SO}}(q_0) = f_{\text{H}} + \log(w_{\text{JE}} e_{\text{JE}} + w_{\text{SE}} e_{\text{SE}}) \tag{5}$$

where,

$$e_{\text{SE}} = \sum_t ||F_{\text{ext}}||^2 \tag{6}$$

and w_{JE} and w_{SE} are the respective weights. The square of external forces (F_{ext}) is summed over all time-steps when they are active.

For effectively computing each point on the pareto-optimal front we run the single objective optimizations using CMA [Hansen 2006] with 16 offsprings per generation and an initial $\sigma = 0.015$. Starting

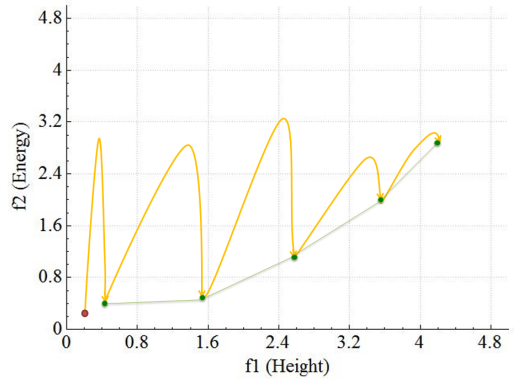


Figure 4: *Binned single objective optimizations (BSOO)*

with the smallest height bin, the optimization for each bin is stopped if the improvement of function value is less than ϵ over a span of 300 generations or the total number of generations produced exceeds 3000. The optimal point for the current bin is used as a starting point for the optimization of the next bin in sequence. This process is stopped after all the bins have been optimized.

The optimization process is illustrated in Figure 4. The task achievement is binned into multiple successive bins of target jump heights. Beginning from an initial state illustrated in red on the left, each of the target height bins is optimized for sequentially. The yellow line provides a schematic illustration of the evolution of solution path over time. Once the optimization has achieved convergence for a particular bin, this sample is then used as a starting point for the optimization for the next bin. In our working example we begin with a jump of low height. This jump is then optimized to achieve the target height of the nearest bin. Once converged, this then seeds the optimization for the next bin and so on until all the bins have been optimizer for.

4.2 Multi-Objective Optimization for Discovering the Pareto-Optimal Front

Multi-objective optimization produces a pareto-optimal set of solutions. Each point in this set represents a particular trade-off between the incommensurable objectives. The animator, system designer, or possibly an automated system can choose the most suitable solution from amongst these trade-offs at run-time. We use a multi-objective optimization for producing energy optimal jumps with a variety of jump heights. The competing objectives in our case are the control effort and the height achieved for the jump. Once computed, any point on the pareto front can be chosen on-the-fly. We build on and adapt the (1+1) Covariance Matrix Adaptation Evolution Strategy for Multi-Objective Optimization ((1+1) MOO-CMA-ES)[Igel et al. 2007] to generate a parameterized family of controllers. We begin by describing the key elements of the original algorithm.

4.2.1 (1+1) CMA-ES

(1+1) CMA-ES is an elitist selection based CMA. One parent produces one offspring with a normal multivariate distribution as $x \sim \mathcal{N}(m, \sigma^2 C)$. The multivariate distribution C and the global step size σ are adapted in each generation of the algorithm. In a single objective setting the success of an offspring depends on the fact that if the offspring has a better fitness value than its parent. This success based update is different from path length control strategy of the standard CMA.

4.2.2 Non-Dominated Sorting

Consider an M dimensional multi-objective optimization problem. Let $f(x)$ be a vector comprising M objective functions.

$$f(x) = (f_1(x), f_2(x), ..., f_M(x)) \tag{7}$$

where, $x \in X$ is the vector of free parameters. The elements of X can be partially ordered using the concept of non-dominance.

$$x \prec x' \iff \forall i \in M : f_i(x) \leq f_i(x') \wedge \exists j : f_j(x) < f_j(x') \tag{8}$$

We can then define a pareto set as follows:

$$\{x \mid \nexists x' \in X : x' \prec x\} \tag{9}$$

If X is the set of all samples then let $ndom_1(X)$ represent the non-dominated set of X and define $dom_0(X) = X$. We can then define the level of dominance in a recursive fashion.

$$dom_l(X) = dom_{l-1}(X) \backslash ndom_l(X), l \in \{1, 2, ...\} \tag{10}$$

$$ndom_l(X) = ndom(dom_{l-1}(X)), l \in \{1, 2, ...\} \tag{11}$$

An element is given a rank l if it belongs to $ndom_l(X)$.

$$rank(x, X) = l \iff x \in ndom_l X \tag{12}$$

The time complexity for this sorting for N elements has been noted to be $O(\mathcal{M}\mathcal{N}^2)$ in [Deb et al. 2002]

4.2.3 Hypervolume Sorting

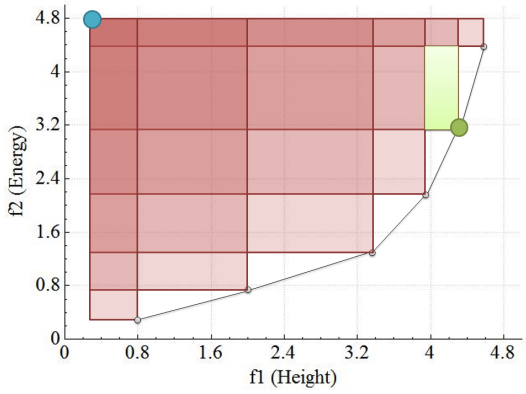


Figure 5: *Hypervolume sorting. The blue point is a reference point chosen to be worse off than all possible samples in all objectives. The red regions shows the area (Lebesque measure in 2D) of rectangles (hypercuboids in 2D) specified by a sample on the pareto-optimal front and the reference point. The green region denotes the area contribution (hypervolume contribution in 2D) of the green non-dominated sample on the pareto-optimal front.*

The hypervolume measure was introduced by [Zitzler and Thiele 1998]. According to [Igel et al. 2007] and [Coello and Lamont 2004] this can be defined as the Lebesque measure ($\wedge$) of union of hypercuboids.

$$S_{x_{ref}}(X) = \wedge(\cup_{x \in ndom(X)}\{F\}) \tag{13}$$

$$F = \{(f_1(x'), f_2(x'), ..., f_M(x')) | x \prec x' \prec x_{ref}\} \tag{14}$$

where x_{ref} is a reference point which is worse than each objective function value. The hypervolume contribution of x is then defined as follows:

$$\Delta_s(x, X) = S_{x_{ref}}(X) - S_{x_{ref}}(X \backslash x) \tag{15}$$

A pictorial representation of the hypervolume measure for two objective functions is provided in Figure 5. Boundary elements are given an infinite value for their hypervolume measure. Boundary elements are identified by the fact that their hypervolume contribution changes by moving the reference point whereas for non-boundary elements the hypervolume contribution remains unchanged. [Igel et al. 2007] then defines the following relation based on this ranking and the non-dominance level :

$$\begin{aligned} x \prec_{s,X} x' \iff & rank(x, X) < rank(x', X) \\ & or \\ & [(rank(x, X) = rank(x', X)) \\ & \wedge \\ & \Delta_s(x, ndom_{rank(x,X)}(X)) > \Delta_s(x', ndom_{rank(x,X)}(X))] \end{aligned} \tag{16}$$

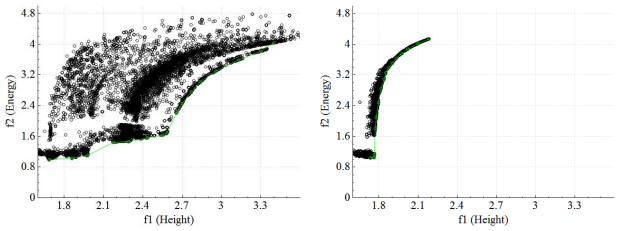
The goal of multi-objective optimization is to find a diverse set of solutions in order to generate a set of good trade-offs between the various incommensurate objectives. To apply (1+1) CMA-ES to multi-objective optimization we maintain a population of size 2μ. This population in the steady state comprises of μ pairs of parents and their offsprings. The μ parents are mutated at each generation to produce an offspring each. The resulting population is then sorted using the criteria define in eq(16). To perform this sorting the population is first sorted using the non-dominance criteria and then the non-dominated subset of the population is sorted using the hypervolume contribution criteria to maximize dispersion. The second sorting criteria is required because as the optimization progresses more samples are in the top-most non-dominated set than the ones in the next level of non-dominated sets.

4.3 Re-sampled Multi Objective Optimization (RMOO)

We now describe a modified version of the standard (1+1) MOO CMA-ES algorithm which is used to effectively optimize a diverse set of solutions automatically spanning natural and supernatural domains on the pareto-optimal front.

4.3.1 Resampling

Jumping in place for an articulated character is a highly constrained problem. Failure can arise from multiple reasons: the character can land incorrectly and fall over, or the character can land outside the landing region. These cases result in large failure penalties of fixed magnitude, f_p, to be returned during the objective function evaluation. An offspring producing such behavior is deemed as being unsuccessful in terms of not producing a better fitness value than its parent. Since (1+1) CMA-ES uses a success based path update rule, each such failure contributes to reduction of the global step size σ over time. This can cause possibly good samples to be discarded from the pareto-optimal front in favor of samples which were 'better' offsprings in recent generations. In the long term this can cause premature convergence. Such issues can be significantly diminished by providing a sampling that is guaranteed to have a minimal degree of success. We found allowing a maximum of 3 bad samples per generation to be the best compromise between computational performance and the quality of the pareto-front obtained. The re-sampling is performed in a multi-threaded fashion with each unsuccessful sample spawning a set of new threads which run the simulations in parallel with other such threads until the given sampling success criterion is met. Figure 6 provides a comparison of strategies with and without the use of resampling to meet the success criterion. The resampling strategy produces a significantly better pareto optimal front at a computational cost. Through our experiments we found that making the success criteria tighter significantly increased the computational cost while the pareto-optimal front generated was quite similar in quality to that produced using the proposed success criteria. On the other hand relaxing the success criteria from its current value degraded the quality of pareto-optimal from significantly. We believe that this success criteria would work well for other controller tasks as well but this has been left for future work.



(a) Re-sampling with a success rate of 13/16

(b) No re-sampling

Figure 6: *Comparison between re-sampling strategies*

4.3.2 Scaling of Fitness Value

We want the character to maximize the utilization of its internal energy for task achievement. If the character relies merely on external forces for task achievement without utilizing its internal energy to a maximum possible extent, it would appear that the character just gets lifted off by an external force while trying to achieve a really high jump. We want the motion to appear as natural as possible even in the supernatural domain and hence we require the character to exert itself as much as possible in the supernatural domain. One way to achieve this is by scaling external force penalty with an appropriate w_{SE}. If w_{SE} is made large enough then the character favors usage of its internal joint torques over external forces towards task achievement. However, since the selection of pareto-optimal front depends on the hypervolume contribution of each sample point on the front, this scaling causes most of the samples in the natural domain to be lost since their hypervolume contribution turns out to be low. This can be seen in the pareto-optimal front obtained by using a large w_{SE} in Figure 7. The character used for this simulation can jump up to nearly 2.3-2.5m high without any help from external forces. But since the hypervolume contribution of sample points in this region is low, only 2 points get picked in this region. The graph has been plotted on a log scale for consistency with the other plots, however the hypervolume contribution is computed on a linear scale. Moreover, if the weight (w_{SE}) is not extremely large (there's a limit to how large the weight can be because any evaluation must not exceed f_p), some points obtained in the possibly natural region still have some supernatural component, albeit a small one. We would like to have zero external force contribution for points which correspond to jump heights in the natural region. We found an objective function of the form $\log(w_{\mathrm{JE}} e_{\mathrm{JE}} + w_{\mathrm{SE}} e_{\mathrm{SE}})$ to perform particularly well. $w_{SE}/w_{JE} = 5000$ was used for all experiments. The log form works well firstly because the spread of fitness values produced is almost uniform across both natural and super-natural domains. Secondly, since w_{SE} is large, a clear de-

marcation is observed between natural and super-natural samples.

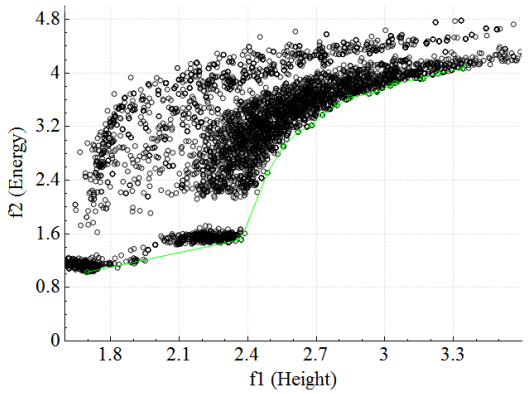


Figure 7: *The effect of a large penalty weight w_{SE} but no scaling on pareto-optimal front distribution. Only two controllers in the natural regime are produced. The controllers on the pareto-optimal front for a height $> 2.4m$ use an external force assist.*

A high level description of the RMOO algorithm is given in Algorithm 1.

5 Results

We apply the algorithms discussed in Section 4 (BSOO and RMOO) to build a family of pareto-optimal controllers for a standing jump motion developed for a physics based character. Figure 8 compares the pareto-optimal front obtained from the BSOO and the RMOO algorithms. The RMOO strategy is able to discover solutions which could not be found with the BSOO algorithm. This was possibly due to the BSOO running into local minima. The solution space topology is better explored through a concerted effort from the multiple (1+1) CMA algorithms running in parallel. Also, RMOO is faster than the BSOO optimization technique by a factor of nearly 2. It took nearly 5-6 hours for a multi-threaded code running on 3.2Ghz 8 core Intel Xeon based machine to generate the results shown in Figure 8 using RMOO with $\mu = 16$. On the other hand it took nearly 13 hours to generate the results using BSOO strategy for 16 samples.

The log scaled effort objective works well in producing a pareto optimal front that spans the natural and supernatural regimes with a reasonable spacing between points on the front. The two regions have different behaviors as can be seen from Figure 8 (a). The usage of external forces in the supernatural region produces a different response of energy fitness with respect to height than that produced in the natural region with no external forces available. The controllers lying in the natural region ($< 2.5m$ for this example) use precisely 0 external forces whereas those in the supernatural region rely on non-zero external forces to achieve the task. As expected, the reliance on external forces increases as the jump height increases. We color code the characters in the video and the illustrations in this paper in order to distinguish the two motion regimes. Brown characters represent natural motions while blue characters represent supernatural motions.

When optimizing within the bins using a single objective function strategy, spatial coherence in the objective function space is not exploited. Naively generating samples using a single objective optimization produces significant redundancy in the sampled set which could have been effectively used to communicate information (solution points) to the neighboring binned optimizations. This is likely

Data: Successful input low height jumping controller, μ
Result: μ sample controllers on the Pareto-Optimal Front
begin
  Initialize μ parents using the parameters from the input controller;
  Evaluate the input controller for objectives f1 and f2 and assign these as the current fitness values to all the parents;
  while *not converged* **do**
    Copy the parameters from each of the μ parents to μ children respectively;
    Mutate the parameters of each of the μ children;
    Evaluate and set the fitness of the μ children in a multi-threaded fashion;
    while *number of bad samples* > 3 **do**
      Assign the threads with copies of unsuccessful parents;
      Mutate the parameters of all copies;
      Perform multi-threaded evaluation of all the copies;
    end
    Perform non-dominated sort;
    Perform hypervolume sort;
    Choose the top μ candidates to be parents for the next generation;
    Update CMA strategy parameters for all parents;
    if *change of fitness for all samples* $< \epsilon$ **then**
      set converged;
    end
  end
end

Algorithm 1: A high level description of RMOO for a working example of in-place jumps

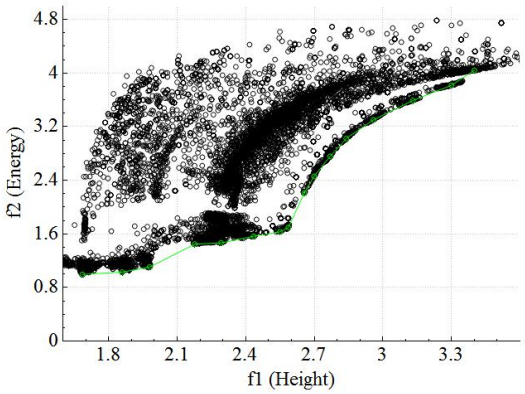
one of the principle reasons for the performance gains shown by the RMOO strategy.

Since, in our framework the character prefers to use internal forces to help achieve the required jump height, even when in the supernatural regime, the take-offs for the supernatural jumps and the highest natural jumps remain very similar in style as seen in Figure 9. As the jump height increases the character exerts more internal energy for task achievement. The internal energy used for a jump should be maximized for the highest possible natural jump. In order to perform supernatural jumps the character should still exert the same amount of internal effort but it can also take help from external forces to achieve an even higher jump. There was no explicit reward term in the optimization to make the take-offs look similar. We tabulate the values of samples on the pareto-front for the two optimization techniques in Tables 2 and 3.

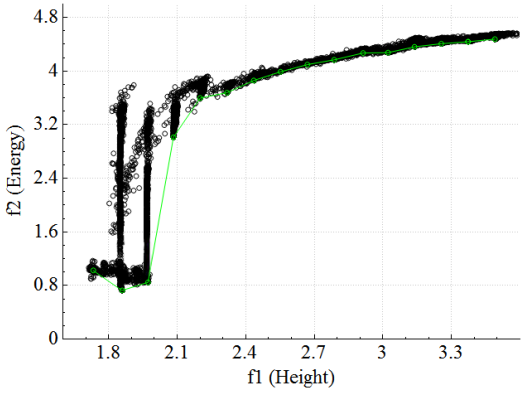
6 Conclusion

A multi-objective framework for optimizing physics-based character animation is an interesting and viable approach for designing controllers. Access to a set of choices on the pareto-optimal front allows the character to perform different motions for different task objectives at run-time. Controllers spanning the entire pareto-optimal front across natural and supernatural regions can be automatically generated from single run of optimization by an appropriate design of objective functions. Similar approach could be applied to other problem domains such as generating energy-optimal locomotion gaits of varying speeds.

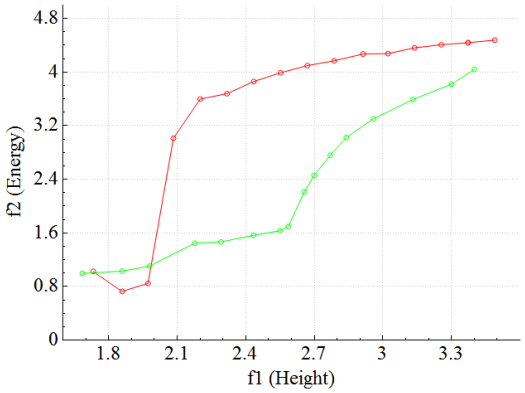
Since the algorithm produces a spread of fitness values for each objective function dimension, care needs to be taken while designing the objective function. Hard penalties muse be used for objective



(a) Pareto-optimal front obtained from RMOO. Green circles connected by lines represents the pareto-optimal front. Black circles are the samples generated during the optimization procedure.

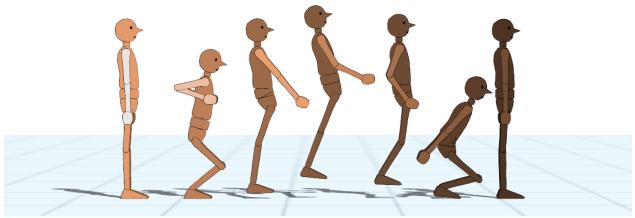


(b) Samples of energy optimal jumps of various heights obtained from BSOO are shown as green circles connected by lines. Black circles are the combined samples generated during each binned optimization procedure.

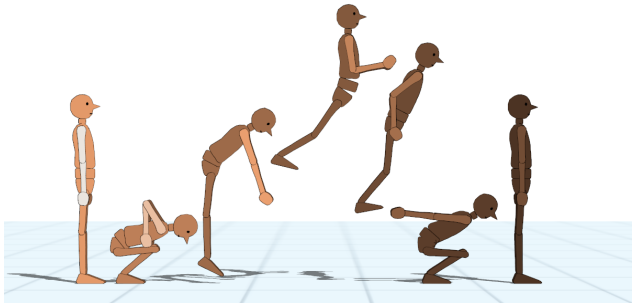


(c) Overlaid results of pareto-optimal front obtained from RMOO (green) and samples generated from BSOO (red)

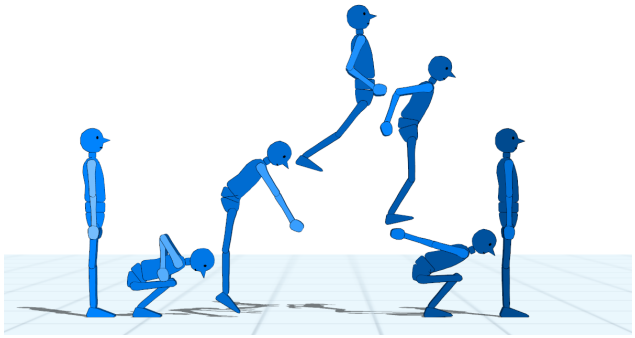
Figure 8: *Plots of samples generated from BSOO and plots of pareto-optimal fronts obtained from RMOO*



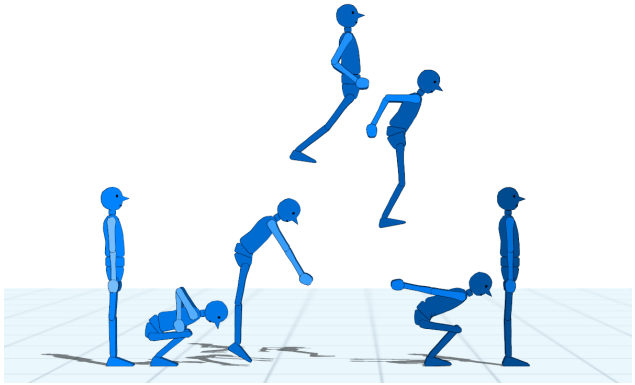
(a) Shortest natural jump



(b) Highest natural jump



(c) Intermediate height supernatural jump



(d) Highest supernatural jump

Figure 9: *Natural and supernatural motions. The snapshots of the motions are created by capturing the pose at the beginning of each phase. The snapshots for each of the standing in-place jumps are offset in x for visualization purposes.*

function terms whose spread is not desired. Although providing hard penalties for objective function terms which are not required in the spread produces problems for the re-sampling process. The samples producing failures for these objective function terms are given a constant failure penalty. This does not provide enough useful information for making the optimum covariance matrix update which adds to the computational cost of the algorithm. As future

sample #	Internal Joint Energy Metric	Super-natural take-off gravity	Super-natural force application time
sample 0	10.953	-9.8	0
sample 1	15.776	-9.8	0
sample 2	19.6615	-9.8	0
sample 3	22.674	-9.8	0
sample 4	26.8889	-9.8	0
sample 5	33.0792	-9.8	0
sample 6	46.7629	-9.8	0
sample 7	52.9262	-8.8	0.64
sample 8	47.2799	-8.374	0.652
sample 9	49.2547	-7.8	0.593
sample 10	49.9069	-7.053	0.591
sample 11	48.9008	-6.08	0.585
sample 12	49.4255	-5.121	0.591
sample 13	52.3201	-4.173	0.610
sample 14	55.3722	-2.921	0.605
sample 15	71.4787	-2.022	0.682

Table 2: *Results using RMOO*

sample #	Internal Joint Energy Metric	Super-natural take-off gravity	Super-natural force application time
sample 0	9.495	-9.8	0
sample 1	5.316	-9.8	0
sample 2	6.715	-9.8	0
sample 3	13.6204	-5.4	0.266
sample 4	12.248	-2.46	0.378
sample 5	16.2731	-2.641	0.474
sample 6	19.5386	-0.442	0.424
sample 7	19.1376	0.2697	0.492
sample 8	21.1071	1.3538	0.513
sample 9	22.8364	1.1774	0.621
sample 10	25.4831	3.2524	0.556
sample 11	26.4307	3.2291	0.566
sample 12	22.0267	2.5725	0.7614
sample 13	27.016	3.55	0.729
sample 14	33.5313	4.2319	0.706
sample 15	40.1909	3.831	0.816

Table 3: *Results using BSOO*

work, we expect that there remain further improvements that could be achieved for computing pareto optimal fronts. The generated motions can be made to look more natural by providing more natural inputs and/or by developing some data driven effort metrics which are more consistent with observed motions.

7 Acknowledgments

We would like to thank NSERC and GRAND for providing the funding for this project. We would also like to thank the anonymous reviewers for their time and comments.

References

AGRAWAL, S., SHEN, S., AND VAN DE PANNE, M. 2013. Diverse motion variations for physics-based character animation. *Symposium on Computer Animation*.

BORNO, M. A., DE LASA, M., AND HERTZMANN, A. 2013. Trajectory optimization for full-body movements with complex contacts. *IEEE Trans. Visualization and Computer Graphics*. in press.

CMLABS. Vortex simulator, v5.1.

COELLO, C., AND LAMONT, G. 2004. *Applications Of Multi-Objective Evolutionary Algorithms*. Advances In Natural Computation. World Scientific Publishing Company Incorporated.

COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2010. Generalized biped walking control. *ACM Transctions on Graphics 29*, 4, Article 130.

COROS, S., KARPATHY, A., JONES, B., REVERET, L., AND VAN DE PANNE, M. 2011. Locomotion skills for simulated quadrupeds. *ACM Transactions on Graphics (TOG) 30*, 4, 59.

DA SILVA, M., ABE, Y., AND POPOVIĆ, J. 2008. Simulation of human motion data using short-horizon model-predictive control. In *Computer Graphics Forum*, vol. 27, Wiley Online Library, 371–380.

DE LASA, M., MORDATCH, I., AND HERTZMANN, A. 2010. Feature-Based Locomotion Controllers. *ACM Transactions on Graphics 29*, 3.

DEB, K., PRATAP, A., AGARWAL, S., AND MEYARIVAN, T. 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on 6*, 2, 182–197.

GEIJTENBEEK, T., AND PRONOST, N. 2012. Interactive Character Animation Using Simulated Physics: A State-of-the-Art Review. *Computer Graphics Forum 31*, 8 (Dec.), 2492–2515.

HANSEN, N. 2006. The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, J. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, Eds. Springer, 75–102.

HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. 1995. Animating human athletics. In *SIGGRAPH '95*, ACM, 71–78.

IGEL, C., HANSEN, N., AND ROTH, S. 2007. Covariance matrix adaptation for multi-objective optimization. *Evol. Comput. 15*, 1 (Mar.), 1–28.

IORIO, A. W., AND LI, X. 2004. Solving rotated multi-objective optimization problems using differential evolution. In *Australian Conference on Artificial Intelligence*, 861–872.

KWON, T., AND HODGINS, J. K. 2010. Control systems for human running using an inverted pendulum model and a reference motion capture sequence. *The ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA 2010)*.

LASZLO, J., VAN DE PANNE, M., AND EUGENE, F. 1996. Limit cycle control and its application to the animation of balancing and walking. In *SIGGRAPH '96*, ACM, 155–162.

LEE, Y., KIM, S., AND LEE, J. 2010. Data-driven biped control. *ACM Trans. Graph. 29* (July), 129:1–129:8.

LIU, C., AND POPOVIĆ, Z. 2002. Synthesis of complex dynamic character motion from simple animations. In *ACM Transactions on Graphics (TOG)*, vol. 21, ACM, 408–416.

LIU, L., YIN, K., VAN DE PANNE, M., SHAO, T., AND XU, W. 2010. Sampling-based contact-rich motion control. *ACM Transctions on Graphics 29*, 4, Article 128.

MACCHIETTO, A., ZORDAN, V., AND SHELTON, C. 2009. Momentum control for balance. In *ACM Transactions on Graphics (TOG)*, vol. 28, ACM, 80.

MUICO, U., LEE, Y., POPOVIĆ, J., AND POPOVIĆ, Z. 2009. Contact-aware nonlinear control of dynamic characters. In *ACM Transactions on Graphics (TOG)*, vol. 28, ACM, 81.

RAIBERT, M. H., AND HODGINS, J. K. 1991. Animation of dynamic legged locomotion. *ACM SIGGRAPH Computer Graphics 25*, 4 (July), 349–358.

SOK, K. W., KIM, M., AND LEE, J. 2007. Simulating biped behaviors from human motion data. *ACM Trans. Graph. 26* (July).

TSAI, Y.-Y., LIN, W.-C., CHENG, K. B., LEE, J., AND LEE, T.-Y. 2010. Real-time physics-based 3d biped character animation using an inverted pendulum model. *IEEE Transactions on Visualization and Computer Graphics 16* (March), 325–337.

VONDRAK, M., SIGAL, L., HODGINS, J., AND JENKINS, O. 2012. Video-based 3d motion capture through biped control. *ACM Transactions on Graphics (TOG) 31*, 4, 27.

WANG, J. M., FLEET, D. J., AND HERTZMANN, A. 2009. Optimizing walking controllers. *ACM Trans. Graph. 28* (December), 168:1–168:8.

WANG, J. M., HAMNER, S. R., DELP, S. L., AND KOLTUN, V. 2012. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Transactions on Graphics (TOG) 31*, 4, 25.

WU, J.-C., AND POPOVIĆ, Z. 2010. Terrain-adaptive bipedal locomotion control. *ACM Transactions on Graphics 29*, 4 (Jul.), 72:1–72:10.

YAMANE, K., AND SOK, K. 2010. Planning and synthesizing superhero motions. *Motion in Games*, 254–265.

YE, Y., AND LIU, C. K. 2010. Optimal feedback control for character animation using an abstract model. *ACM Trans. Graph. 29* (July), 74:1–74:9.

YIN, K. K., LOKEN, K., AND VAN DE PANNE, M. 2007. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics 26*, 3, 105.

ZITZLER, E., AND THIELE, L. 1998. Multiobjective optimization using evolutionary algorithms a comparative case study. 292–301.