Visualizing Movement Control Optimization Landscapes

PERTTU HÄMÄLÄINEN, Department of Computer Science, Aalto University, Finland
JUUSO TOIKKA, Department of Computer Science, Aalto University, Finland
C. KAREN LIU, Department of Computer Science, Stanford University, USA

Fig. 1. Visualizing the effect of different problem modifications on the trajectory optimization landscape of an inverted pendulum, using a trajectory length of 100 timesteps, i.e., a 100-dimensional search space. See section 3 for details. From left to right, 1) “baseline” version where the pendulum is actuated with torques and using squared costs as the objective, 2) converting the costs $c_t$ of timestep $t$ to rewards through exponentiation as $r_t = e^{-c_t}$, 3) the same with early termination, which effectively removes local optima, and 4) same as the baseline, but parameterizing the control as a sequence of target angles, which makes the optimization considerably more separable and well-conditioned.

A large body of animation research focuses on optimization of movement control, either as action sequences or policy parameters. However, as closed-form expressions of the objective functions are often not available, our understanding of the optimization problems is limited. Building on recent work on analyzing neural network training, we contribute novel visualizations of high-dimensional control optimization landscapes; this yields insights into why control optimization is hard and why common practices like early termination and spline-based action parameterizations make optimization easier. For example, our experiments show how trajectory optimization can become increasingly ill-conditioned with longer trajectories, but parameterizing control as partial target states—e.g., target angles converted to torques using a PD-controller—can act as an efficient preconditioner. Both our visualizations and quantitative empirical data also indicate that neural network policy optimization scales better than trajectory optimization for long planning horizons. Our work advances the understanding of movement optimization and our visualizations should also provide value in educational use.

CCS Concepts: • Human-centered computing → Graph drawings; • Theory of computation → Nonconvex optimization; • Computing methodologies → Physical simulation

Additional Key Words and Phrases: Animation, movement synthesis, trajectory optimization, policy optimization, control optimization

1 INTRODUCTION

Much of computer animation research formulates animation as an optimization problem. It has been shown that complex movements can emerge from minimizing a cost function that measures the divergence from movement goals, such as moving to a specific pose or location while minimizing effort. In principle, this holds the promise of elevating an animator to the role of a choreographer, directing virtual actors and stuntmen through the definition of movement goals. However, solving the optimization problems can be hard in practice. It can require hours or even days of computing time, which is highly undesirable for interactive applications and the rapid iteration of movement goals; defining the goals can be a non-trivial design problem in itself, requiring multiple attempts to produce a desired aesthetic result.

Optimization problems can be divided into the four classes of increasing difficulty illustrated in Figure 2:

- **Convex and well-conditioned** Convexity refers to the shape of the isocontours of the objective function, or level sets in a general $d$-dimensional case. In the ideal well-conditioned case, the isocontours are spherical, and simple gradient descent recovers a direct path to the optimum.
- **Convex and ill-conditioned** In ill-conditioned optimization, the isocontours are elongated instead of spherical. The gradient—coinciding with isocontour normals—no longer points towards the optimum, and numerical optimization may require more iterations.
- **Non-convex and unimodal** Non-convexity tends to make optimization even harder, but numerical optimization usually still converges if there are no local optima to distract it.
- **Non-convex and multimodal** In this problem class, the landscape has local optima which can attract optimization. Gradient-free, sampling-based approaches like CMA-ES—common in animation research—may still find the global optimum [Hansen and Kern 2004], but this may be computationally expensive.
Unfortunately, movement optimization can easily fall into this class, e.g. due to the discontinuities caused by colliding objects, and multiple options for going around obstacles [Hämäläinen et al. 2014, 2015].

Landscape visualizations like those in Figure 2 provide useful intuitions of optimization problems, and can help in reformulating a problem into a more tractable form. In general, one would like the objective function to be more convex, well-conditioned, and unimodal. We know that problem modifications such as the choice of action visualization approach of [Li et al. 2018] can be applied in the domain of movement optimization. Figure 1 provides examples of this, using the same basis vectors in all the plots for more consistent comparisons. Figure 3 shows the same objective function with different random basis vectors. Furthermore, we use the visualizations to investigate the following research questions:

- What is the effect of the number of timesteps – i.e., the planning horizon – on the optimization landscape? (Section 4)
- What is the effect of the choice of action space on the optimization landscape? (Section 5)
- What is the effect of converting instantaneous costs into rewards through exponentiation? (Section 6)
- What is the effect of early termination of movement trajectories or episodes, e.g., when deviating from a target state? (Section 7)
- Do the visualizations predict actual optimization performance, and generalize from simple to complex problems? (Sections 8 and 9).

Additionally, Section 10 provides a more theoretical investigation of the reliability and limitations of random 2D slice visualizations. We conclude that such visualization is a useful tool for diagnosing problems; the somewhat low sensitivity is compensated by high specificity. Our visualizations also explain why movement optimization best practices such as early termination and parameterizing actions as target angles work so well, which should make our work useful in teaching computer animation and movement optimization.

2 RELATED WORK

Spacetime optimization. Much of the earlier work on animation as optimization focused on extensions of the seminal work on spacetime optimization by Witkin and Kass [Cohen 1992; Fang and Pollard 2003; Safonova et al. 2004; Wampler and Popović 2009; Witkin and Kass 1988], where the optimized variables included the root position and rotation as well as joint rotations for each animation frame. However, the synthesized motions were limited by the need for prior knowledge of contact information, such as when and which body parts should make contact with the ground. This limitation was overcome by [Mordatch et al. 2012], who introduced auxiliary optimized variables that specify the contact information. However, the number of colliding body parts was still limited.

Animation as simulation control. In recent years, the focus of research has shifted towards animation as a simulation control problem. Typically, one optimizes simulation control parameters such as time-varying actuation torques of character joints, and an off-the-shelf physics simulator is used to realize the movement. While spacetime optimization can be performed with gradient-based optimization methods like Sequential Quadratic Programming [Witkin and Kass 1988] or L-BFGS [Mordatch et al. 2012], simulation control is commonly approached with sampling-based, gradient-free optimization due to non-differentiable dynamics and/or multimodality [Hämäläinen et al. 2014, 2015; Liu et al. 2010]. This is also what our work focuses on; it remains as future work to extend our visualizations to analyzing spacetime optimization.

Trajectory and policy optimization. Two main classes of approaches include trajectory and policy optimization. In trajectory optimization, one optimizes the time-varying control parameters directly, which can be done both offline [Al Borno et al. 2013a; Naderi et al.
Although it is technically trivial to extend this to higher-dimensional cases, players perform the optimization aided by predictive optimization through a conversion to an interactive game or puzzle [Li et al. 2018]. There are also examples of visualizing movement optimization from the path taken during optimization [Goodfellow et al. 2014; Smith and Topin 2017], or visualizing in a plane determined from the found optimum [Dinh et al. 2017; Goodfellow et al. 2014; Ngo and Marks 1993] or online, while the character moves and acts [Hämäläinen et al. 2014, 2015; Tassa et al. 2012]. In policy optimization, one optimizes the parameters of a policy function such as a neural network that maps character state to (approximately) optimal control, typically independent of the current simulation time. This can be done both using neuroevolution [Geijtenbeek et al. 2013; Such et al. 2017] or Reinforcement Learning (RL), which has recently proven powerful even with complex humanoid movements [Peng et al. 2018; Schulman et al. 2017]. Unfortunately policy optimization/learning can be computationally expensive with large neural networks, and may require careful curriculum design [Yu et al. 2018]. On the other hand, it can produce controllers that require orders of magnitude less computing resources after training, compared to using trajectory optimization to solve each required movement in an interactive application such as a video game. Trajectory and policy optimization approaches can also be combined [Levine and Koltun 2013; Mordatch and Todorov 2014; Rajamäki and Hämäläinen 2018], which allows one to adjust the trade-off between training time and runtime expenses. In this paper, we provide analyses of both trajectory and policy optimization landscapes.

Visualizing optimization. Many optimization visualizations are problem-specific, utilizing the semantics of optimized parameters [Jones 1994]. Visualization is also used for letting a user interact and inform optimization [Meignan et al. 2015]; this, however, falls outside the scope of this paper. Considering non-interactive, generic methods applicable to continuous-valued optimization, landscape visualizations like the ones in Figure 2 are a standard textbook method. Although it is technically trivial to extend this to higher-dimensional problems by visualizing the objective function on a random plane (a 2D subspace), [Li et al. 2018] only recently demonstrated that such random slices can provide meaningful insights and, likewise, have some predictive power on the difficulty of very high-dimensional optimization. Inspired by [Li et al. 2018], we test the random slice visualization approach in a new domain, and also provide additional analyses of the method’s reliability and limitations. Other common methods for visualizing high-dimensional optimization include graphing the objective function along a straight line from the initial point to the found optimum [Dinh et al. 2017; Goodfellow et al. 2014; Keskar et al. 2016; Smith and Topin 2017], or visualizing in a plane determined from the path taken during optimization [Goodfellow et al. 2014]. There are also examples of visualizing movement optimization through a conversion to an interactive game or puzzle; in this case, players perform the optimization aided by predictive visualizations of how different actions affect the simulation state [Hämäläinen et al. 2017].

3 TEST PROBLEM: INVERTED PENDULUM BALANCING

This section describes the inverted pendulum balancing problem that is used throughout Sections 4-8, before testing the visualization approach on the more complex simulated humanoid of Section 9. The pendulum is depicted in Figure 4. Although it is simple, it offers the following benefits for analyzing movement optimization:

- In the trajectory optimization case, we know the true optimum. As the pendulum dynamics are differentiable, we can also compute the Hessian and its eigenvalues for further analysis. We implement this using Autograd [Maclaurin et al. 2015].

- In the policy optimization case, we can use a simple P-controller as the parameteric policy, which admits visualizing the full optimization landscape instead of only low-dimensional slices.

The dynamics governing the angle $\alpha_t$ and angular velocity $\omega_t$ of the pendulum at timestep $t$ are implemented as:

$$\omega_{t+1} = \omega_t + \delta (\tau_t + 0.5 l g \sin(\alpha_{t-1})),$$

$$\alpha_{t+1} = \alpha_t + \delta \omega_t,$$

where $\delta$, $l$, and $g$ are the simulation timestep, pendulum length, and force induced by gravity, respectively. We use $\delta = 0.1$, $l = 0.2$, and $g = 9.81$.

3.1 Trajectory Optimization

Most of our trajectory optimization visualizations are generated from the problem of balancing a simple inverted pendulum, starting from an upward position such that the optimal torque sequence...

Fig. 3. Visualizing the inverted pendulum trajectory optimization objective of Equation 3 with different random basis vectors and $T = 100$. Although the plots are not exactly similar, they all exhibit the same overall structure, i.e., multimodality and an elongated, ill-conditioned optimum.

Fig. 4. The inverted pendulum model. The force exerted by gravity is denoted by $g$, and $\alpha$ denotes the angular deviation from an upright position.

\[
\delta, l, \text{and } g \text{ are the simulation timestep, pendulum length, and force induced by gravity, respectively. We use } \delta = 0.1, l = 0.2, \text{ and } g = 9.81.
\]
The reward formulation has been recently used with stellar results in the policy optimization of complex humanoid movements [Peng et al. 2018]. Exponentiation is also used in framing optimal control as estimation [Todorov 2008, 2009].

3.2 Policy Optimization

In the case of policy optimization, we use the same pendulum simulation, with a minor adjustment. Instead of directly optimizing control torques, we use a policy \( \tau_t = \pi_\theta(s_t) \), parametrized by \( \theta \), where \( s \) denotes pendulum state. The optimization objective is to either minimize the expected trajectory cost, or maximize the expected reward, assuming that each trajectory or “episode” is started from a random initial state. As closed-form expressions for the expectations are not available, we replace them by averages. These are computed from 10 episodes, each started from a different initial pendulum angle.

In all the pendulum policy optimization visualizations, we use a simple P-controller as the policy:

\[
\pi_\theta(s_t) = \theta \tau_t. 
\] (5)

The benefit of this formulation is that we only have a single policy parameter to optimize; thus, we can visualize the full objective function, shown in Figure 5. Despite the simplicity, this provides a multimodal optimization problem with properties similar to more complex problems. As shown in Figure 5, there is a global optimum at approximately \( \theta = -0.1 \), and also a false optimum near \( \theta = 0 \). At the false optimum, the action cost is minimized simply by letting the pendulum hang downwards. In a real-world case like controlling a simulated humanoid, the false optimum corresponds to resting on the ground with zero effort; readers familiar with humanoid control probably know that such a behavior is easy to elicit by having too large an effort cost.

The success of local gradient-based optimization depends on which cost landscape \( \mathcal{R} \) using exponentiated costs as

\[
\mathcal{R} = \sum_{t=1}^{T} (e^{-\alpha t^2} + we^{-\tau^2}). 
\] (4)

The trajectory optimization landscapes of Figure 6 are augmented with visualizations of the Hessian matrices of the cost function \( C \), and also a false optimum near \( \theta = 0 \). At the false optimum, the action cost is minimized simply by letting the pendulum hang downwards. In a real-world case like controlling a simulated humanoid, the false optimum corresponds to resting on the ground with zero effort; readers familiar with humanoid control probably know that such a behavior is easy to elicit by having too large an effort cost.

The reward formulation has been recently used with stellar results in the policy optimization of complex humanoid movements [Peng et al. 2018]. Exponentiation is also used in framing optimal control as estimation [Todorov 2008, 2009].
The Hessian of the cost function can then be expressed as:

\[ \nabla_{\tau} \nabla_{\alpha} C_{t} = 2 \sum_{t=1}^{T} (\nabla_{\tau} a_t)(\nabla_{\alpha} a_t)^T + \alpha_t \frac{\partial^2 C}{\partial \alpha^2} + 2w d. \]  

(7)

The last term of Equation 7 is a diagonal matrix that does not depend on \( T \), while the first term results in off-diagonal elements accumulated over time. As \( T \) grows larger, the first term dominates the second one, hence the decrease in separability (i.e., the fading diagonal shown in Figure 6).

The increase of the condition number follows from the causality of the dynamics—future actions do not affect past states. This means that when \( t = 1 \), the first term of Equation 7 is a \( T \times T \) matrix that has only one nonzero element at the upper-left corner, while, when \( t = T \), the nonzero elements occupy the entire matrix. Summing up all the matrices from \( t = 1 \) to \( T = T \), the first term of the Hessian will have larger values toward the upper-left corner, as evidenced by the bright areas in Figure 6. The longer \( T \) is, the more disparate the upper-left and bottom-right corners become, leading to larger condition numbers.

Although it is well known that the difficulty of solving a trajectory optimization increases proportionally with the length of the planning horizon, our visualization shows that the difficulty is not solely caused by the increase in the size of the problem, but also by the increasingly ill-conditioned and non-separable objective function. Section 5 explains how the choice of action parameters can act as an efficient preconditioner.

4.2 Policy Optimization

In policy optimization, the optimized parameters have no similar dependency to the horizon \( T \), as the effect of each policy parameter on the objective is averaged over all states. The separability of policy parameters depends on the function representation of the policy. For example, if the policy is represented as a multi-layer nonlinear neural network, the Hessian of the objective function is bound to be inseparable, as the neuron weights across layers are multiplied with each other when passing data through the network.

However, a longer horizon in policy optimization can induce a multimodal landscape in the object function (Figure 7). An explanation is that a longer time budget enables strategies that are not possible when the horizon is shorter. The “mollification” of the landscape at shorter horizons explains why policy optimization may benefit from a curriculum in which the planning horizon is gradually increased. The OpenAI Five Dota 2 bots provide a recent impressive example of this, gradually increasing the \( \gamma \) parameter of Proximal Policy Optimization [Schulman et al. 2017] during training.

5 EFFECT OF THE CHOICE OF ACTION SPACE

In the previous section, we saw that minimizing effort as the sum of squared actions gives rise to a strong constant diagonal in the Hessian, which in principle leads to better conditioned optimization. Parameterizing actions as torques, and having a squared torque cost term, is also common in continuous control benchmark problems such as OpenAI Gym MuJoCo [Brockman et al. 2016]. However, [Peng and van de Panne 2017] showed that parameterizing actions as target joint angles can make policy optimization more effective. The target poses that the policy outputs are converted to joint torques, typically using a P- or PD-controller. Such pose-based control is also common in earlier work on trajectory optimization [Al Borno et al. 2013b; Hämäläinen et al. 2014, 2015; Rajamäki and Hämäläinen 2017], often claimed to give better results than optimizing raw torques; despite this, comprehensive comparisons of control parameterizations are rare.

Figure 8 shows that using actions as target angles \( \alpha \) does indeed scale better to long horizons. The strong constant diagonal of the Hessian does not fade out as much, and the condition number \( \kappa \) grows much slower with larger \( T \). We compute the torques using a PD-controller: \( \tau = k_p(\bar{\alpha} - \alpha) + k_d \dot{\alpha} \). Using \( k_p = 1 \) and \( k_d = -1 \). The optimal \( \bar{\alpha} \) sequence is all zeros, as the pendulum starts from an upright position with zero velocity.

An explanation for the effectiveness of the target angle parameterization is that actions represent (partial) target states. This makes the optimization of state cost or reward terms more separable and well-conditioned, and the Hessian closer to a scaled identity matrix. The optimal action at each timestep is more independent of preceding actions; a reasonable strategy is to always drive the pendulum
where the dependency to past states and actions is primarily due to
2014; Naderi et al. 2017]. The discussion above provides a strong
ing P or PD-controllers [Al Borno et al. 2013b; Hämäläinen et al.
controlled character over time, which are then implemented us-

ted action sequences, in effect, the control points of a spline can be thought
of as a shorter sequence of higher-level actions, each of which
defines the instantaneous actions for multiple timesteps.
Naturally, using splines has aesthetic motivations as well, as they
result in smoother motion with less frame-by-frame noise.
Experimental results comparing spline-based optimization with
direct optimization of action sequences are provided in Section 9.

5.1 Pose splines
In many papers, long action sequences are parameterized as “pose
splines”, i.e. parametric curves that define the target pose of the
controlled character over time, which are then implemented using
P or PD-controllers [Al Borno et al. 2013b; Hämäläinen et al.
2014; Naderi et al. 2017]. The discussion above provides a strong
motivation for this, as such parameterization achieves two things at once:

• Better separability and conditioning of the problem due to
  the choice of action space.
• Improved avoidance of ill-conditioning with long action se-
  quences; in effect, the control points of a spline can be thought
  of as a shorter sequence of higher-level actions, each of which
defines the instantaneous actions for multiple timesteps.

6 EFFECT OF USING REWARDS INSTEAD OF COSTS

In Section 8, we shall see that the reward function can indeed
result in worse optimization performance. However, the next section
discusses how using rewards instead of costs can be more desirable
when combined with the technique of early termination.

\[ \alpha_t = \alpha_{t-1} + \delta \omega_{t-1} + \delta^2 (\tau_t + 0.5 l g \sin(\alpha_{t-1})), \]  
\[ \alpha_t = (1 - \delta^2) \alpha_{t-1} + (\delta - \delta^2) \omega_{t-1} + \delta^2 (\tau_t + 0.5 l g \sin(\alpha_{t-1})). \]  

Figure 9 shows the inverted pendulum trajectory optimization
landscape using the reward function of Equation 4. Comparing this
to Figure 6, one notices the following:

• With large \( T \), the landscape structures are essentially the
  same for both the cost function (Equation 3) and the reward
  function (Equation 4), with an elongated optimum and some
  local optima. The Hessian and \( \kappa(T) \) at the optimal point are
  the same in both cases. Thus, in principle, the reward function
  should be as easy or hard to optimize as the cost function.
• On the other hand, at \( T = 10 \), the reward landscape shows
  some additional non-convexity. This suggests that in prac-
tice, the reward function may cause a performance hit in
  optimization. Section 8 provides evidence of this.

The main difference of the functions is that the sum of exponen-
tiated costs is more tolerant to temporary deviations. A sum-of-
squares cost function heavily penalizes the cost being large in even
a single timestep, whereas the exponentiation clamps the rewards
in the range \([0, 1]\). In principle, an agent could exploit the reward
formulation by only focusing on some reward terms. We interpret
the ridges in Figure 9 \((T = 10)\) as manifestations of this.

In Section 8, we shall see that the reward function can indeed
result in worse optimization performance. However, the next section
discusses how using rewards instead of costs can be more desirable
when combined with the technique of early termination.
7 EFFECT OF EARLY TERMINATION

Standard continuous control policy optimization benchmark tasks [Brockman et al. 2016; Tassa et al. 2018] utilize early termination of simulated episodes; this means that if the agent deviates from some desired region of the state space, e.g. a bipedal agent loses balance, the state is considered a terminal one; the agent stops receiving any rewards, and is reset to an initial state. Typically, termination greatly speeds up policy optimization, e.g. [Peng et al. 2018]. As far as we know, early termination has not been utilized in trajectory optimization, but there are no obstacles for this.

Figures 10 and 11 show the effect of terminating trajectories and episodes if $|\theta| > 2.0$, i.e. if the pendulum deviates significantly from the desired upright pose. From the figures, one can gain two key insights:

- **Termination can greatly improve landscape convexity by removing false optima.** The pendulum cannot receive rewards from the local optimum of hanging downwards if termination prevents it from experiencing the corresponding states.
- **If the rewards are not strictly non-negative, new false optima are introduced to the landscape.** Naturally, if the agent is experiencing costs or negative rewards, a good strategy may be to terminate episodes as early as possible, which is what happens at the $\theta = 0.5$ local optimum in Figure 11 (second subfigure from left, when termination is used with cost minimization). The problem can be mitigated by adding a termination penalty to the cost of the terminal simulation step, or a so-called alive bonus for all non-terminal states. This yields a clearly more convex landscape.

Although an alive bonus is used in many papers and benchmark tasks [Brockman et al. 2016; Rajeswaran et al. 2016; Yu et al. 2018], we feel the danger of combining termination and negative rewards is not emphasized enough in previous literature. Our visualization highlights that the early termination can be a double-edged sword—it can be harmful when the reward function consists of a mixture of positive and negative terms that are not well-balanced. Our visualization technique suggests that a good default strategy may be to use a combination of termination and rewards instead of costs. This way, one does not need to fine-tune the termination penalty or alive bonus. Furthermore, converting costs to rewards through exponentiation limits the results to the range $[0, 1]$, which is beneficial for most RL algorithms that feature some form of value function learning with neural networks. On the other hand, as shown in the next section, it can result in slower convergence.

8 DO VISUALIZATIONS PREDICT OPTIMIZATION PERFORMANCE?

Figure 12 tests how well the visualizations of the previous sections predict actual optimization performance in the inverted pendulum trajectory optimization. The figure compares quadratic costs to exponentiated rewards, and actions parameterized as torques to target angles implemented using a P-controller as in Section 5. To keep the figure readable, we did not include curves with and without termination; this does not make a big difference in the simple pendulum problem, and termination is further investigated in the next section. All the optimizations were performed using CMA-ES [Hansen and Ostermeier 2001], which is common in animation research, and is known to perform well with multimodal optimization tasks. A population size of 100 was used. To allow comparing both costs and rewards, progress is graphed as the Euclidean distance from the true optimum.

The results indicate that parameterizing actions as target angles is considerably more efficient, as predicted by the visualizations of Section 5. The exponential transform of costs to rewards degrades performance, in line with the observations of Section 6.

9 GENERALIZABILITY TO MORE COMPLEX AGENTS

This section tests the generalizability and usefulness of the visualization approach with a more complex agent, using both policy optimization and trajectory optimization. We use the 3D humanoid locomotion test from the Unity Machine Learning Agents framework [Juliani et al. 2018] shown in Figure 13. The test is designed for policy optimization; we modify it to also support trajectory optimization by starting each episode/trajectory from a fixed initial state, without randomization. We have also modified the code and environment to be fully deterministic, i.e., non-reproducible simulation is not corrupting the landscape visualizations.

The action space is 39-dimensional, with actions defined as both joint target angles, as well as maximum torques that the joint motors are allowed to use for reaching their targets. We optimize with planning horizons of 1, 3, 5, and 10 seconds, resulting in 585, 1755, 2925, and 5850 optimized variables. The actions define target angles and maximum torques for the 16 joints of the humanoid. We visualize the landscapes around the optimum found in each test. We use a simulation timestep of 1/75 seconds, with control actions repeated for 5 timesteps, i.e., taking 15 actions per second.

As detailed below in Sections 9.1-9.4, the results support our earlier findings:

- **2D visualization slices reveal that trajectory optimization is highly multimodal, with optima that become narrower as the length of the planning horizon increases.**
- **As hypothesized in Section 5.1, utilizing a spline parameterization results in a more well-behaving landscape.**
- **Termination based on agent state reduces local optima in both trajectory and policy optimization.**
- **Policy optimization with neural network policies scales better for long planning horizons, with the landscape remaining essentially unchanged as the planning horizon grows.** Notably, **policy optimization was more efficient than optimizing a single long trajectory**, even though our policy network has over 2M parameters to optimize, i.e., orders of magnitude more.

9.1 Trajectory Optimization

For trajectory optimization we use the recent highly scalable CMA-ES variant called LM-MA-ES [Loshchilov et al. 2018]—as the physics simulator is not differentiable, gradient-based methods are not applicable. We visualize the landscapes around the optimum found by LM-MA-ES. Similar to CMA-ES, LM-MA-ES is a quasi-parameter-free method, typically only requiring adjustments to the iteration sampling budget (population size): this is increased from the recommended value for more difficult problems. The recommended
Fig. 10. Effect of early termination on inverted pendulum trajectory optimization landscape. Termination without an alive bonus increases multimodality of cost minimization, but makes reward maximization more convex.

Fig. 11. Effect of termination policy optimization. In cost minimization, termination creates a local minimum at $\theta = 0.5$, which drives the pendulum to termination to avoid accumulating more costs. Termination removes local optima when combined with an alive bonus or using rewards instead of costs.

Fig. 12. Inverted pendulum trajectory optimization results, plotted as the mean of 10 runs of CMA-ES. We compare both cost minimization and reward maximization with actions parameterized both as torques and target angles. As predicted by our visualizations, the angle parameterization scales better for large $T$, and the reward maximization is less efficient.

Fig. 13. 3D humanoid locomotion test from Unity Machine Learning Agents framework.

Fig. 14 shows the landscapes around the found local optimum for each tested planning horizon. Although there is a clear optimum in the center, the rest of the landscape is noisy and ill-conditioned. This is exacerbated with the longer planning horizons.

Figure 14 also shows that termination helps remove local optima, and produces a smoother landscape. In the humanoid locomotion test, termination takes place when a body part other than the feet touches the ground.

9.2 Trajectory Optimization with Splines
In an effort to better handle long planning horizons, we tested trajectory optimization with spline-based parameterization. Instead of setting target angles and maximum torques for joints once every 5 timesteps, we interpolated the actions for each timestep using Catmull-Rom splines. The control points of the splines were optimized using LM-MA-ES, akin to the method presented by Hämäläinen et al. [2014] for online optimization. Figure 15 shows how this improves the landscapes, with reduced noise and gentler slopes towards the optimum.
To provide a more direct comparison to optimizing action sequences without splines, Figure 16 shows the non-spline trajectory optimization landscape around the action sequences resulting from the optimized splines of Figure 15. Without the spline parametrization, the 3s and 5s landscapes degenerate, becoming noisier and more ill-conditioned.

Although trajectory optimization with splines results in cleaner landscapes and qualitatively better and smoother movement than non-spline trajectory optimization, finding good long trajectories required hours of CPU time. This motivated us to also test policy optimization, as explained below.

9.3 Policy Optimization

We trained a neural network policy for solving the humanoid locomotion task using Proximal Policy Optimization (PPO) [Schulman et al. 2017] and different planning horizons. PPO utilizes episodic experience collection, i.e., the agent is started from some initial state, and explores states and actions until a terminal state, or maximum episode length, is reached. We used the planning horizon as the episode length.

The resulting landscapes, shown in Figure 17, show a significant improvement in convexity, sphericity, and unimodality. The landscape also remains essentially unchanged with a longer planning horizon.

9.4 Scalability of Trajectory and Policy Optimization

The policy optimization visualizations suggest that as trajectory lengths increase, policy optimization should be more efficient than trajectory optimization. Figure 18 provides evidence supporting this hypothesis. It compares optimization progress with different
planning horizons and the three optimization strategies: LM-MA-ES, LM-MA-ES with splines, and PPO. With a trajectory length of 5 seconds, PPO already shows improved performance over the other methods, and the advantage is dramatically larger with 10 second trajectories. This confirms our hypothesis, but comes with a caveat: with a neural network policy, there is more overhead per simulation step. Optimizing for 5 million timesteps with PPO was approximately 4 times slower than with LM-MA-ES in our tests, when measured in wall-clock time, using a single 4-core computer.

Interestingly, although the spline landscapes look better, spline trajectory optimization results in slightly lower rewards with a given simulation budget. A plausible explanation for this is that the Unity locomotion test has a very simple reward function that trajectory optimization can exploit with unnatural and jerky movements, as shown on the supplemental video\(^2\) at 03:01. From a pure reward maximization perspective, a spline parameterization that enforces a degree of smoothness may not be ideal and one may expect more gains if the reward function favors smooth movement. In general, best results are achieved when the action parameterization induces a useful prior for the optimization task.

10 PROPERTIES AND LIMITATIONS OF RANDOM 2D SLICE VISUALIZATIONS

So far, we have provided many examples of random 2D slice visualizations of high-dimensional objective functions. We have also demonstrated that such visualizations have predictive power regarding the difficulty of optimization. However, as information is obviously lost in only evaluating the objective along a 2D slice, this section provides further analysis of the limitations of the approach.

10.1 2D Visualizations are Optimistic About Ill-conditioning

Consider the following cost function:

\[
 f(x) = \sum_{i=1}^{k} x_i^2 + \epsilon \sum_{i=k+1}^{d} x_i^2 \]

where \(\epsilon\) is a small constant, i.e., \(f(x)\) mostly depends only on the first \(k\) optimized variables. \(x_k\) denotes the projection of \(x\) into the subspace of the first \(k\) dimensions, \(x_k\) denotes the projection into the remaining dimensions. The Hessian of \(f(x)\) is diagonal, containing the curvatures along the unit vectors as the diagonal elements. Curvature along the first \(k\) unit vectors equals 2 and curvature along the rest of the dimensions is \(2\epsilon\). Thus, if \(k \neq 0\) and \(k \neq d\), the condition number \(\kappa = 1/\epsilon\).

Geometrically, the isosurfaces of \(f(x)\) are \(n\)-spheres elongated by a factor of \(1/\sqrt{\epsilon}\) along the last \(d-k\) dimensions. The visualized 2D isocontours correspond to the intersections of the isosurfaces with the visualization plane. This is illustrated in Figure 19 and on the supplemental video for \(d = 3\).

[Fig. 18. Best trajectory or episode reward as a function of timesteps simulated during optimization, with different planning horizons and optimization approaches. The graphs show the mean and standard deviation of 5 independent optimization runs. Policy optimization with PPO scales significantly better for long planning horizons.]

[Fig. 19. The isosurfaces, random visualization planes, and 2D visualization isocontours for Equation 11 in the case of \(d = 3\).]

Investigating Figure 19 reveals a basic property of the 2D visualizations: with the convex quadratic objective of Equation 11, the elongation of the 2D isocontours is less than or equal to the true elongation of the isosurfaces. In other words, \(\kappa_{2D} \leq \kappa\).

This property follows from the isocontours corresponding to planar intersections of the isosurfaces. First, as illustrated in the middle of Figure 19, it is possible to rotate the visualization plane such that the isocontours display less elongation. Second, the isocontours cannot display more than the real elongation; the visualized elongation is at maximum when one plane basis vector aligns with a direction of high elongation (vertical axis in the middle of Figure 19), and the other basis vector aligns with a direction of low elongation, in which case the isocontours display the correct \(\kappa_{2D} = \kappa = 1/\epsilon\).

10.2 2D Visualizations Show Ill-conditioning More Accurately With Low Intrinsic Dimensionality

It turns out that visualization accuracy depends the intrinsic dimensionality \(k\). To analyze this, let us consider the extreme case of \(\epsilon = 0\), i.e.,

\[
 f(x) = f(x_k) = ||x_k||^2. \]

Figure 20 shows the contour plots of random 2D slices with different \(k\) and \(d\). With low \(k\), the visualized ill-conditioning is more accurate, independent of full problem dimensionality \(d\). Deriving a closed-form expression of \(\kappa_{2D}\) as a function of \(d\) and \(k\) is beyond

---

\(^2\)https://youtu.be/6CXfzeO0604
the scope of this paper. However, as shown below, \( k = 1 \) and \( k = d \) result in the correct \( \kappa_{2D} = \infty \) and \( \kappa_{2D} = 1 \), respectively.

Let \( u, v \in \mathbb{R}^d \) denote the slice basis vectors, with orthogonality \( u^T v = 0 \). On the visualization plane, \( x = p u + p v = [u \ v] p \), where \( p \) denotes the 2D position on the plane. Similarly, \( x_{\cdot k} = [u_k \ v_k] p \), and the objective can be expressed as:

\[
\begin{align*}
    f(x_{\cdot k}) &= ||[u_k v_k] p||^2 \\
    &= p^T [u_k^T u_k \ u_k^T v_k \ v_k^T u_k \ v_k^T v_k] p \\
    &= p^T A p.
\end{align*}
\]

Because \( A \) is symmetric, the Hessian of the quadratic form w.r.t. \( p \) is:

\[
H(f(x_{\cdot k})) = A + A^T = 2A.
\]

The condition number \( \kappa_{2D} = \kappa(H(f(x_{\cdot k}))) = \kappa(A) \), as the condition number is invariant to scaling the Hessian by a constant.

With \( k=1 \), \( u_k = u_1, v_k = v_1 \) are scalars, and the determinant becomes zero:

\[
\begin{align*}
    \det(A) &= u_k^T u_k v_k^T v_k - u_k^T v_k v_k^T u_k \\
    &= u_1 u_1 v_1 v_1 - u_1 v_1 v_1 u_1 = 0.
\end{align*}
\]

This indicates at least one zero eigenvalue and, since \( A \) is not a null matrix, some eigenvalue must also be nonzero, i.e., \( \kappa(A) = \max(eig(A))/\min(eig(A)) = \infty \).

When \( k \) grows from 1 to \( d \), \( u_k, v_k \) gradually become closer to \( u, v \), i.e., unit-length and orthogonal. Thus, the off-diagonal elements become zero and \( A \) becomes the identity matrix, with \( \kappa(A) = 1 \).

Although the quadratic \( f(x) \) was chosen to be separable for easier mathematical analysis, the result generalizes to the arbitrarily rotated case.

### 10.3 2D Visualizations Can Be Optimistic About Multimodality

Figure 21 shows contour plots of random visualization slices with different dimensionality \( d \), using two multimodal test functions. Rastrigin’s function is a standard multimodal optimization test function with the global minimum at the origin and infinitely many local minima:

\[
f_{\text{Rastrigin}}(x) = 10d + \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i)]
\]

Additionally, we use the following bimodal function:

\[
f_{\text{Bimodal}}(x) = \varepsilon^{-\frac{1}{2}} ||x-1||^2 + 0.8\varepsilon^{-\frac{1}{2}} ||x+1||^2,
\]

where \( \varepsilon \) denotes a vector of ones. We visualize this function both around the origin and around the dominant mode at 1.

Figure 21 reveals two key insights:

- Multimodality becomes less apparent with increasing dimensionality \( d \). The exception is the first row, where visualized multimodality only depends on plane rotation independent of \( d \). This is because the visualization plane intersects the origin; in this case, the optima are always equally far from the plane and thus have similar influence on the visualized \( f_{\text{Bimodal}}(x) \). However, when the plane intersects the optimum at 1, the other optimum tends to lie increasingly far from the plane with increasing \( d \), having a negligible effect on the visualization.

- The visualization of Rastrigin’s function illustrates how landscape features may scale differently with dimensionality. Rastrigin’s central mode becomes more dominant with large \( d \). At the bottom of Figure 21, we demonstrate how this can be compensated by omitting the unit-length normalization of the slice basis vectors, and instead normalizing them to the mean of their sampled lengths. Each basis vector element is sampled uniformly in the range \([-1, 1]\).

### 10.4 If 2D Visualizations Show Problems, There Really Are Problems

**Visualized ill-conditioning is real.** The results above indicate that 2D visualizations have limited sensitivity as a diagnostic tool for detecting ill-conditioning and multimodality. Fortunately, \( \kappa_{2D} \leq \kappa \) also means that the visualizations have high specificity, i.e. if the 2D isocontours are elongated, the problem is indeed ill-conditioned.

**Visualized non-convexity indicates real non-convexity.** For a convex unimodal objective, the 2D visualization is likewise convex and unimodal. This follows from the intersection of two convex subsets of \( \mathbb{R}^d \), i.e., the visualization plane and the volume enclosed by the corresponding isosurface. However, other non-convexity can be confused with multimodality. Consider a curved, banana-shaped 3D isosurface. It is possible to intersect this with a plane such that the resulting isocontours comprise two ellipses.
We have presented several novel visualizations of continuous control actions. These visualizations have pedagogical value in educating new researchers and practitioners. For example, we show how termination based on agent state objectives tend to be optimistic about both ill-conditioning and multimodality. However, this limitation is mitigated by the visualizations not showing illusory non-convexity or ill-conditioning. In other words, as a diagnostic tool for understanding why optimization does not converge, or progresses slowly. We have also presented an analysis of the limitations of the random 2D slice visualizations.

We acknowledge that some of our results—e.g., the efficiency of episode termination—are already known to experienced readers. We do, however, provide novel visual evidence of the underlying reasons; for example, we show how termination based on agent state removes local optima in the space of optimized actions or policy parameters. This contributes to the understanding of movement optimization, and, as representative images are known to increase understanding and recall [Carney and Levin 2002], it should also have pedagogical value in educating new researchers and practitioners.

To conclude, the key insights from our work can be summarized as:

- Random 2D slice visualizations are useful in analyzing high-dimensional movement optimization landscapes, and can predict movement optimization efficiency.
- The curse of dimensionality hits trajectory optimization hard, as it can become increasingly ill-conditioned with longer planning horizons. Policy optimization scales better in this regard. Perhaps counterintuitively, optimizing a neural network policy can be more efficient than optimizing a single action trajectory with orders of magnitude less parameters.
- Parameterizing actions as [partial] target states—e.g., target angles that the character’s joints are driven towards—is strongly motivated, as opposed to optimizing raw control torques. It can make trajectory optimization more well-conditioned and separable.
- Combining the two points above, one can explain the power of the common practice of optimizing splines that define time-varying target poses; pose parameterization leads to more well-conditioned optimization, and the spline control points define a shorter sequence of macro actions, which further counteracts ill-conditioning caused by sequence length. However, the smoothness constraints that splines impose on movements may not be ideal for all reward functions.
- Using early termination appears strongly motivated, as it typically results in a more convex landscape in both trajectory and policy optimization. However, combining termination with costs or negative rewards is dangerous, which our visualizations clearly illustrate.

In our future work, we aim to investigate the parameterization of actions as target states for complex controlled agents with unactuated roots. We hypothesize that this can be implemented for both trajectory and policy optimization, using a general-purpose neural network controller trained for reaching the target states.

11 CONCLUSION

We have presented several novel visualizations of continuous control trajectory and policy optimization landscapes, demonstrating the usefulness of the random 2D slice visualization approach of [Li et al. 2018] in this domain. The visualizations provide new intuitions of movement optimization problems and explain why common best practices are powerful. The visualization approach can be used as a diagnostic tool for understanding why optimization does not converge, or progresses slowly. We have also presented an analysis of the limitations of the random 2D slice visualizations.

10.5 Summary of Limitations

In summary, random 2D visualization slices of high-dimensional objectives tend to be optimistic about both ill-conditioning and multimodality. However, this limitation is mitigated by the visualizations not showing illusory non-convexity or ill-conditioning. In other words, as a diagnostic tool for detecting problems, random 2D visualizations have low sensitivity compensated by high specificity.
Computer-Human Interaction in Play. ACM, 497–505.

Petruu Hämäläinen, Joose Rajamäki, and C Karen Liu. 2015. Online Control of Simulated Humans Using Particle Belief Propagation. ACM Transactions on Graphics 34, 4 (2015), 81.

Nikolaus Hansen. 2016. The CMA evolution strategy: A tutorial. arXiv preprint arXiv:1604.00722 (2016).

Nikolaus Hansen and Stefan Kern. 2004. Evaluating the CMA evolution strategy on multimodal test functions. In International Conference on Parallel Problem Solving from Nature. Springer, 282–291.

Nikolaus Hansen and Andreass Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. Evolutionary computation 9, 2 (2001), 159–195.

Christopher V Jones. 1994. Visualization and optimization. ORSA Journal on Computing 6, 3 (1994), 221–257.

Arthur Juliani, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. 2018. Unity: A general platform for intelligent agents. arXiv preprint arXiv:1809.02627 (2018).

Nith Shurish Keskar, Dheevatsa Madireddy, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tuck Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. arXiv preprint arXiv:1609.04836 (2016).

Sergey Levine and Vladlen Koltun. 2013. Guided policy search. In International Conference on Machine Learning. 1–9.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. In Advances in Neural Information Processing Systems. 6389–6399.

Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. 2010. Sampling-Based Contact-Rich Motion Control. ACM Transactions on Graphics 29, 4, Article 128 (July 2010), 10 pages. https://doi.org/10.1145/1778765.1778865

Ilya Loshchilov, Tobias Glassmachers, and Hans-Georg Beyrer. 2018. Large Scale Black-box Optimization by Limited-Memory Matrix Adaptation. IEEE Transactions on Evolutionary Computation (2018).

Dougal Maclaurin, David Duvenaud, and Ryan P Adams. 2015. Autograd: Effortless gradients in numpmy. In ICLR 2015 AutoML Workshop.

David Meignan, Sigrid Knust, Jean-Marc Frayret, Gilles Pesant, and Nicolas Gaud. 2015. A review and taxonomy of interactive optimization methods in operations research. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 3 (2015), 17.

Igor Mordatch and Emo Todorov. 2014. Combining the benefits of function approximation and trajectory optimization. In Robotics: Science and Systems. 5–32.

Igor Mordatch, Emanuel Todorov, and Zoran Popović. 2012. Discovery of Complex Behaviors Through Contact-invariant Optimization. ACM Trans. Graph. 31, 4 (July 2012), 43:1–43:8. https://doi.org/10.1145/2185520.2185539

Kourosh Naderi, Joose Rajamaki, and Perttu Hamalainen. 2017. Discovering and synthesizing humanoid climbing movements. ACM Transactions on Graphics (TOG) 36, 4 (2017), 43.

J. Thomas Ngo and Joe Marks. 1993. Spacetime Constraints Revisited. In Proc. SIGGRAPH ’93. ACM, New York, NY, USA, 343–350. https://doi.org/10.1145/166117.166160

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. arXiv preprint arXiv:1804.02717 (2018).

Xue Bin Peng and Michiel van de Panne. 2017. Learning locomotion skills using deeprl: does the choice of action space matter? In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. ACM, 12.

Joose Rajamaki and Petruu Hämäläinen. 2017. Augmenting Sampling Based Controllers with Machine Learning. In Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA ’17). ACM, New York, NY, USA, Article 11, 9 pages. https://doi.org/10.1145/3099564.3099579

Joose Julius Rajamäki and Perttu Hamalainen. 2018. Continuous control monte carlo tree search informed by multiple experts. IEEE transactions on visualization and computer graphics (2018).

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. 2016. Epopt: Learning robust neural network policies using model ensembles. arXiv preprint arXiv:1610.01283 (2016).

Alla Safonova, Jessica K. Hodgings, and Nancy S Pollard. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. ACM Trans. Graph. 23, 3 (2004), 514–521. https://doi.org/10.1145/10155754

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017).

Leslie N Smith and Nicholay Topin. 2017. Exploring loss function topology with cyclical learning rates. arXiv preprint arXiv:1702.04283 (2017).

Felipe Petroski Such, Vashishat Madhavan, Edosardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. 2017. Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arXiv preprint arXiv:1712.06567 (2017).

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. 2018. DeepMind Control Suite. arXiv preprint arXiv:1801.06690 (2018).