CHRONOSRL: TEMPORAL GEOMETRY FOR SELF-SUPERVISED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

A goal that is close in space can be far away in time. Obstacles, terrain, and the agent's own capabilities determine how long it takes to get there. Yet, critics in contrastive and survival reinforcement learning do not measure the distances in their representation space in units of time. We therefore introduce ChronoSRL, which gives the critic's embeddings an explicit temporal geometry. The distance between state–action and goal embeddings is trained to match the time that the agent takes to reach the goal (goal-reaching time), while goals that were not reached, and goals from other trajectories, are pushed at least one discount horizon away. Furthermore, reaching a goal quickly once does not mean that reaching it is reliable in general, so the policy should not follow the temporal distance directly. Instead, we build on survival reinforcement learning and predict from our temporal embeddings not only the full distribution of goal-reaching times but also the time spent near the goal. Thereby, the policy is trained to favor actions that reach the goal sooner and more reliably and that keep the agent near it. ChronoSRL learns faster and reaches higher performance than contrastive, action-chunked contrastive, and survival reinforcement learning baselines on seven standard locomotion and navigation benchmarks, even with much smaller networks. To test the limits of self-supervised reinforcement learning, we introduce velocity tracking, goal-position reaching, and box climbing tasks with a quadruped robot in a realistic sim-to-real locomotion setup, and show how the shaping terms that are typical for robotics can be naturally incorporated into our framework. ChronoSRL is the only one of the tested self-supervised reinforcement learning methods that learns to stay at the commanded velocities and goal positions, and climbs the highest boxes.

1 INTRODUCTION

When an agent interacts with its environment in Reinforcement Learning (RL), it collects trajectories that might or might not end up at its intended goals. Nevertheless, each trajectory can be used as a source of information that shows how to reach the various states visited along the way. Self-supervised RL relabels these states as goals, turning every rollout of the agent into training data for a rich goal-conditioned policy without the need to design a hand-crafted reward for each goal (Andrychowicz et al., 2017; Eysenbach et al., 2022). This makes learning possible even when the agent struggles to reach its intended goal at the start of the learning process.

For relabeled goals to be useful, the agent must learn how to relate states, and therefore goals, to one another. Simply relying on the physical distance between states can easily lead to local optima, as often seen in maze-like environments. The time needed to reach a particular state provides a more accurate measure of its distance, capturing the dynamics of the environment, like obstacles, and the agent's capabilities. Naturally, every state in a trajectory provides information about how long it takes to reach any other subsequent state, simply by its time index in the trajectory. A representation of states that encodes this temporal distance can then allow the agent to predict how long it will take to reach different goals from any given state.

We introduce ChronoSRL to make the temporal distance an explicit part of the representation space in self-supervised goal-conditioned RL. Building on temporal distance learning (Hartikainen et al., 2020; Mendonca et al., 2021; Bohlinger, 2023), we train the distance between state–action and goal

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

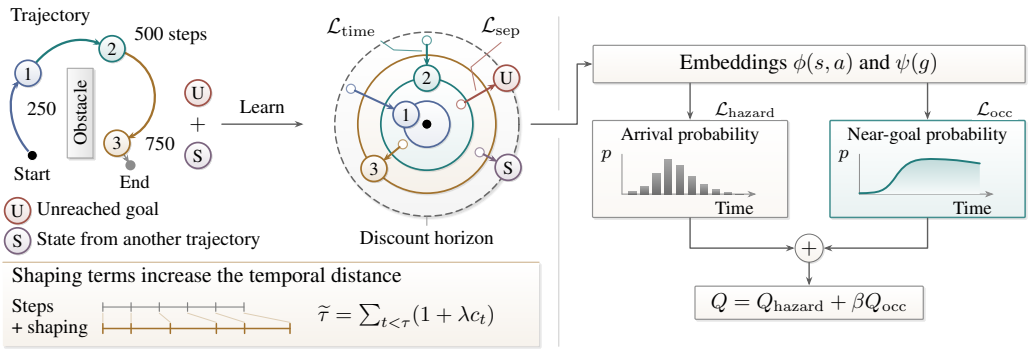


Figure 1: ChronoSRL learns temporal geometry from trajectories. State 3 is physically closer to the start than state 2 but takes longer to reach. The time loss trains embedding distances to match goal-reaching times, while the separation loss pushes unreached goals and states from other trajectories at least one discount horizon away. The shared embeddings also predict the distribution over goal-reaching times with a hazard head and the time spent near the goal with an occupancy head. Both heads form the value that trains the policy, and shaping terms can add per-step costs to the distance.

embeddings on the time taken to reach the goal (goal-reaching time). Short goal-reaching times give small target distances in representation space, and long times give larger ones. We put these targets on a shared scale by measuring time in units of the discount horizon. For a measured goal-reaching time τ and discount factor γ, the target distance between the embeddings is $-\tau \log \gamma$, such that the objectives for representation learning and control use the same time scale. Goals that were not reached and states from other trajectories can also be used to gain even more supervision signal, by treating them as at least one full discount horizon away.

Giving the distance between embeddings a temporal meaning does not necessarily make them sufficient for choosing actions. A risky action can reach a goal quickly on a single successful attempt and still fail most of the time. An agent can also learn to overshoot and reach a goal region briefly and fail to stay there consistently. Therefore, learning must account for both the uncertainty in reaching the goal and what happens after reaching it. Thus, we build on the idea of Survival Reinforcement Learning (SRL) (Nguimatsia-Tiofack et al., 2026), and use our temporal embeddings to predict both the full distribution over goal-reaching times and the time spent near the goal. The policy is trained on the value obtained from these predictions, and learns to favor actions that reach the goal sooner and more reliably and keep the agent nearby.

We evaluate ChronoSRL on seven standard locomotion and navigation environments (Bortkiewicz et al., 2025) and compare it to Contrastive Reinforcement Learning (CRL) (Wang et al., 2026), Action-Chunked Contrastive Reinforcement Learning (AC-CRL) (Korniak et al., 2026), and SRL (Nguimatsia-Tiofack et al., 2026). ChronoSRL learns faster than all three baselines across network depths and typically reaches the final performance of the strongest baseline with less than a third of the training budget. Already with eight layers, it learns faster than any baseline with up to 64 layers. Beyond these standard benchmarks, we evaluate the limits of ChronoSRL and the baseline methods in more challenging and realistic robotics scenarios. We build on the typical sim-to-real pipeline for legged robot locomotion and design velocity tracking, goal-position reaching, and box climbing tasks for the Unitree Go2 quadruped robot. We show how the shaping terms that are necessary and typical for robotics can be naturally incorporated into the self-supervised RL framework. ChronoSRL is the only one of these methods that learns to stay at the commanded velocities and goal positions, and climbs the highest boxes.

2 Related Work

Goal-conditioned RL trains a single policy to reach many different goals (Liu et al., 2022). Kaelbling (1993) already showed how each observed transition of an agent could be used to update its estimates of the expected remaining steps for multiple goals. Afterwards, universal value function approximators were introduced, generalizing over multiple goals through a shared function approx-

imator (Schaul et al., 2015). Such goal-conditioned value functions can also learn from trajectories that do not reach their intended goal, since they still reach other states. Hindsight experience replay uses these achieved states as alternative goals for learning the value function (Andrychowicz et al., 2017), while the relabeled trajectories can also be used as demonstrations for training the policy directly with supervised learning (Ghosh et al., 2021).

The same trajectories can also be used to learn representations that predict which states the agent is likely to visit. Successor representations describe these predictions through discounted future visitations (Dayan, 1993; Barreto et al., 2017). One way to learn such predictions is to distinguish future states from randomly sampled goals, following the ideas of noise-contrastive estimation and contrastive predictive coding (Gutmann & Hyvärinen, 2010; Oord et al., 2018). C-learning and CRL use this classification problem to learn goal-conditioned value functions (Eysenbach et al., 2021; 2022). Temporal-difference updates extend this approach by combining information across trajectories (Zheng et al., 2024b), while improvements to network architectures and training make CRL effective on offline robotics data (Zheng et al., 2024a). Alongside recent progress in scaling value functions in RL (Nauman et al., 2024; Lee et al., 2025; Palenicek et al., 2026), Wang et al. (2026) show that increasing network depth allows CRL to continuously scale and improve its performance. Building on the idea of action chunking (Zhao et al., 2023; Li et al., 2026), Korniak et al. (2026) extend CRL to action sequences and improve performance even further.

While CRL uses InfoNCE to distinguish future goals from randomly sampled goals (Oord et al., 2018; Eysenbach et al., 2022; Wang et al., 2026), the same trajectories also provide step counts between states, which temporal-distance methods use to learn distances and construct rewards for goal reaching (Florensa et al., 2019; Venkattaramanujam et al., 2019; Hartikainen et al., 2020; Mendonca et al., 2021; Bohlinger, 2023). Each state can be paired with later states, providing targets over many distances within a trajectory. Such distances can depend on the direction, since reaching a state may take longer than returning from it. Quasimetric models capture this asymmetry while satisfying the triangle inequality (Wang & Isola, 2022; Liu et al., 2023). Quasimetric RL builds on these models with an objective for learning optimal goal-reaching distances (Wang et al., 2023; Zheng et al., 2026), and Myers et al. (2024) show that a temporal quasimetric can also be derived from contrastive successor representations. In offline and unsupervised goal-conditioned RL, temporal distances can be used to structure the latent space (Park et al., 2024a;b; Bae et al., 2024; Myers et al., 2026; Boock et al., 2026) and guide the selection of subgoals (Park et al., 2023). ChronoSRL introduces its temporal structure through a learning objective rather than enforcing a quasimetric distance function. We train distances between state–action and goal embeddings to match observed goal-reaching times in units of the discount horizon. Whereas other methods use the learned distance itself as the value or reward that the policy maximizes, ChronoSRL uses it only to shape the representation of its critic, whose value estimate is then a combination of the predictions of the goal-reaching time and goal-region occupancy.

Goal-conditioned value functions can also be learned by predicting when a goal will be reached. A trajectory that ends at the intended goal provides a goal-reaching time, while an attempt that ends before reaching it provides a lower bound on this time. Survival analysis uses these measured times and lower bounds to learn a distribution over possible goal-reaching times (Kaplan & Meier, 1958; Cox, 1972). Building on neural survival models (Katzman et al., 2018; Kvamme et al., 2019; Gensheimer & Narasimhan, 2019; Lee et al., 2018), SRL predicts this distribution for each state–action–goal triplet and derives a discounted value from it (Tiofack et al., 2026; Nguimatsia-Tiofack et al., 2026). Since reaching a goal once does not ensure that the agent stays there, Nguimatsia-Tiofack et al. (2026) introduce a dwell-time objective that requires the agent to remain at the goal for a consecutive period of time. ChronoSRL builds on this idea by directly predicting the goal occupancy, which is the expected time spent near the goal. However, only reaching and remaining at a goal does not say anything about how the agent got there. When applying self-supervised RL algorithms to robotics tasks that should be transferable to the real world, it is important to ensure that the learned policies produce smooth trajectories or are influenced by other relevant constraints, such as energy consumption. Such reward shaping terms help the real-world transfer and can also improve the exploration and learning efficiency of the agent (Rudin et al., 2022; Bohlinger et al., 2024). We incorporate these terms as per-step costs in the temporal targets, such that both the learned geometry and the arrival and occupancy predictions account for them.

3 TEMPORAL GEOMETRY FOR SELF-SUPERVISED RL

We propose to give the critic in self-supervised RL a representation in which distances measure time. The core idea is shown in Figure 1. Instead of learning the critic's embeddings only as features for its value prediction, we train the distance between state–action and goal embeddings to match the goal-reaching times observed in the agent's own trajectories. Building on SRL (Nguimatsia-Tiofack et al., 2026), the same embeddings then predict when the goal will be reached and how much time the agent will spend near it. The policy is trained on the value combined from both these predictions.

3.1 BUILDING ON SURVIVAL REINFORCEMENT LEARNING

We consider goal-conditioned RL with states s, actions a, goals g, and a discount factor γ, where the agent reaches a goal once the goal-relevant part of its state, such as its position, lies within a distance ε of g, which defines the goal region $\mathcal{B}_\varepsilon(g)$. Following AC-CRL, the policy $\pi(a \mid s, g)$ outputs a chunk of actions that is executed open-loop, and we write a for such a chunk, while all times count individual environment steps. We use chunks of two actions, as they worked best for ChronoSRL, compared to the three actions of AC-CRL. The critic learns from random batches of training examples, drawn from the trajectories that the agent collects with this policy. Each training example pairs a stored state s and a chunk a with a goal g, which hindsight relabeling can take from a state that the agent reached later in the same trajectory. The time indices of the trajectory from s onward then provide the goal-reaching time τ, which is the number of steps until it first enters $\mathcal{B}_\varepsilon(g)$, with $\tau = 0$ if the goal is already reached. If the goal is not reached within an observation window of W steps, we only know that $\tau \geq W$, and survival analysis calls such an example censored.

We build on SRL, which learns a goal-conditioned value from all of these examples. Its critic encodes the state and the action chunk as an embedding $\phi_\theta(s, a)$ and the goal as an embedding $\psi_\theta(g)$, where θ denotes all parameters of the critic. Based on both embeddings, a hazard head $h_\theta(t \mid s, a, g)$ predicts the probability of reaching the goal at step t, given that it has not been reached before. The survival probability that the goal has still not been reached at step t is then $S_\theta(t \mid s, a, g) = \prod_{u=0}^{t} \big(1 - h_\theta(u \mid s, a, g)\big)$. The hazard head is trained with the negative log-likelihood $\mathcal{L}_{\text{hazard}}$ of all examples, including the censored ones (see Appendix B.3). If episodes ended at the goal, the value of an action would be the negative discounted time spent waiting for it,

$$Q_{\text{hazard}}(s, a, g) = -\sum_{t=0}^{W-1} \gamma^t \, S_\theta(t \mid s, a, g). \tag{1}$$

Since, in the environments we consider, episodes do not end immediately upon reaching the goal, SRL relabels sequences of k consecutive goals, such that reaching the goal requires staying near it for k steps. In this critic, nothing relates the distance between the two embeddings to the time it takes to reach the goal, although this time is exactly what the critic predicts.

3.2 DISTANCES IN UNITS OF TIME

We measure the distance between the two embeddings as $d_\theta(s, a, g) = \|\phi_\theta(s, a) - \psi_\theta(g)\|_2$. A goal reached after τ steps is discounted by $\gamma^\tau = e^{-\kappa\tau}$ with $\kappa = -\log\gamma$. We use the negative log of this discount, $\kappa\tau$, as the target distance, such that a goal at distance d is discounted by exactly e^{-d}. The distance thereby measures time in units of the discount horizon $1/\kappa \approx 1/(1-\gamma)$, which is about 1000 steps for $\gamma = 0.999$. We fit the distances to all measured goal-reaching times with the time loss

$$\mathcal{L}_{\text{time}} = \mathbb{E}_{(s,a,g,\tau)\sim\mathcal{D}_+}\Big[\rho\big(d_\theta(s, a, g) - \kappa\tau\big)\Big], \tag{2}$$

where ρ is the Huber loss with a threshold of one discount horizon and $\mathcal{D}_+$ contains the examples with a measured $\tau \geq 1$, since goals that are already reached are left to the hazard head.

The time loss only constrains pairs that a trajectory connects, so the encoders could still place unrelated states and goals close to each other. Therefore, we also use the pairs without a measured goal-reaching time, marked U and S in Figure 1, which are the censored examples and the cross-batch pairs of each state–action with the goals of all other examples in the batch, excluding goals within ε of its own goal. For these pairs, we only know, or assume, that reaching the goal takes at

least W steps, so we penalize only distances below κW, with a linear version of the margin loss of contrastive metric learning (Hadsell et al., 2006), which we call the separation loss,

$$\mathcal{L}_{\text{sep}} = \frac{1}{|\mathcal{N}|} \sum_{(i,j)\in\mathcal{N}} \big[\kappa W - d_\theta(s_i, a_i, g_j)\big]_+, \tag{3}$$

where $[x]_+ = \max(x, 0)$, $\mathcal{N}$ contains the censored examples and the cross-batch pairs of a batch, and with $\gamma = 0.999$ and $W = 1000$, the margin is almost exactly one discount horizon. Unlike the censored examples, the cross-batch pairs never enter $\mathcal{L}_{\text{hazard}}$, since no trajectory shows that their goals are at least W steps away, and some may in fact be closer (see Appendix B.2).

3.3 From temporal geometry to policy learning

It is tempting to simply convert the distance to a goal into its discount, e^{-d}, and use it as the value that the policy maximizes. However, the time loss only sees reached goals, so two actions that reach the same goal after the same number of steps have the same target distance, even if one succeeds far more often than the other (see Appendix A for a formal argument). The distance also ignores what happens after the goal is reached. Therefore, we use the temporal embeddings to predict when the goal is reached and how long the agent stays near it.

For the first prediction, we keep the hazard head of SRL on top of the temporal embeddings. Since it is also trained on the censored examples, it captures how likely the goal is reached at all. It receives both embeddings as input rather than only their distance, so pairs at a similar temporal distance can still differ in how reliably their goal is reached. For the second prediction, we add an occupancy head, which predicts the probability $O_t = O_\theta(t \mid s, a, g)$ that the agent is in the goal region t steps later, and train it on the occupancies o_t observed in the rollout trajectories. After an environment termination, for example a fall, the observed occupancy is zero for the rest of the window. To reward being close to the center of the goal region, we average the occupancy over the goal regions with radii ε, $\varepsilon/2$, and $\varepsilon/4$ (see Appendix B.4). The occupancy head's loss and discounted value are

$$\mathcal{L}_{\text{occ}} = -\mathbb{E}\Big[\sum_t o_t \log O_t + (1 - o_t)\log(1 - O_t)\Big] \quad \text{and} \quad Q_{\text{occ}}(s, a, g) = \sum_{t=1}^{W-1} \gamma^t O_t. \tag{4}$$

Up to normalization, Q_{occ} is the discounted goal occupancy that CRL learns as its value (Eysenbach et al., 2022), but predicted per future step and fit directly to the observed occupancies. Unlike Q_{hazard}, which only depends on the first arrival, it distinguishes staying at the goal from passing through it or falling after arriving.

Both value estimates, Q_{hazard} and Q_{occ}, count discounted time steps, the steps spent waiting for the goal and the steps spent near it, so the policy maximizes their weighted sum with an entropy bonus, as in Soft Actor-Critic (Haarnoja et al., 2018),

$$J(\pi) = \mathbb{E}_{(s,g)\sim\mathcal{D},\, a\sim\pi(\cdot|s,g)}\big[Q_{\text{hazard}}(s, a, g) + \beta\, Q_{\text{occ}}(s, a, g) - \alpha \log \pi(a \mid s, g)\big], \tag{5}$$

where $\mathcal{D}$ is the replay buffer and α the entropy coefficient. The weight β is the fraction of examples in the batch whose episode terminates within the window, so the policy follows the occupancy value less when the agent rarely falls and more when it falls more often (see Appendix B.5).

Finally, the critic minimizes the sum $\mathcal{L}_{\text{hazard}} + \mathcal{L}_{\text{occ}} + \mathcal{L}_{\text{time}} + \mathcal{L}_{\text{sep}}$ with equal coefficients, and all four losses update the same encoders (see Algorithm 1). The time and separation losses, which we call the geometric losses, add no parameters, and without them, our critic reduces to a survival critic as in Nguimatsia-Tiofack et al. (2026), with an occupancy head and action chunks.

3.4 Shaping terms as additional time costs

Policies that control robots need to act smoothly and reach their goals in a way that humans prefer, which typical reward-based RL encodes as reward shaping terms (Rudin et al., 2022; Bohlinger et al., 2024). Self-supervised RL has no reward terms, but all targets of our critic count time steps, so we simply express these terms as additional time costs. For a per-step cost $c_t \geq 0$ and a weight $\lambda \geq 0$, we replace the time τ in our temporal targets, including the censoring, by the shaped time $\widetilde{\tau} = \tau + \lambda \sum_{t<\tau} c_t$, so every step with a nonzero cost counts as more than one step. Two arrivals after the same number of steps can then differ in distance and value if one accumulates more cost. The policy objective stays unchanged, and $\lambda = 0$ recovers the unshaped step counts (see Appendix B.6).

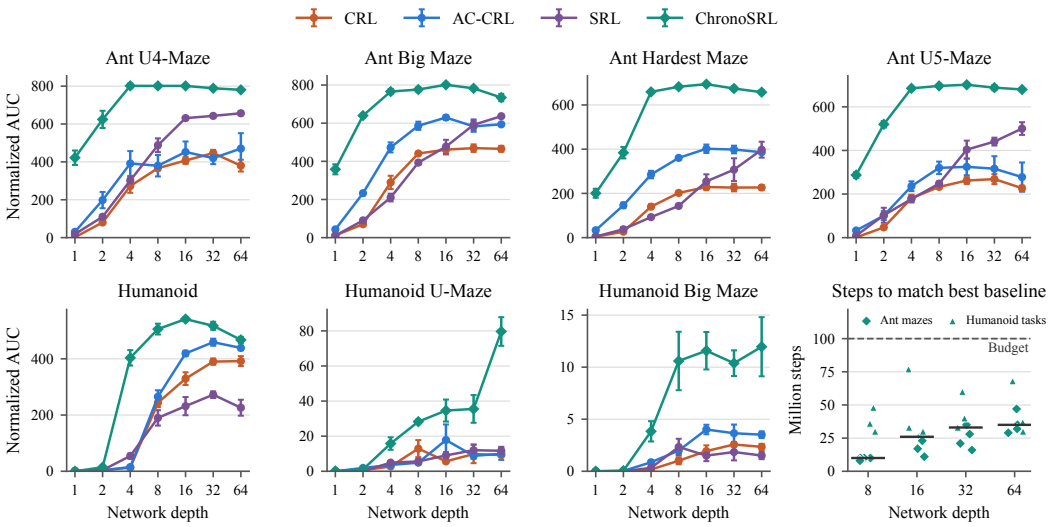


Figure 2: ChronoSRL learns faster and achieves higher final performance than all baselines across network depths. The first seven panels show the mean normalized AUC over four seeds with standard errors, and the last panel shows the steps that ChronoSRL needs to reach the final ToG of the strongest baseline at the same depth.

4 Experiments

We evaluate ChronoSRL on the seven locomotion and navigation tasks that Wang et al. (2026) use in JaxGCRL (Bortkiewicz et al., 2025). Four of these tasks are ant mazes of different sizes and layouts (U4-Maze, Big Maze, Hardest Maze, and U5-Maze), and in the other three, a humanoid has to walk to its goal, either in the open or through a maze (Humanoid, Humanoid U-Maze, and Humanoid Big Maze). Our baselines are the self-supervised methods that have been scaled to these tasks. We run CRL (Wang et al., 2026), AC-CRL (Korniak et al., 2026), and SRL (Nguimatsia-Tiofack et al., 2026) with their published hyperparameters. Since network depth is the main axis along which these methods scale, we train all four methods at depths from 1 to 64, with four seeds per setting and 100 million environment steps per run. We note that this budget is smaller than the budgets that Wang et al. (2026) use for the three hardest mazes (see Appendix C.3). We measure performance by the Time at Goal (ToG), which counts the steps that the agent spends in the goal region during an evaluation episode of 1000 steps. We summarize each learning curve by its final value and by its area (AUC), normalized by the budget (see Appendix D).

4.1 ChronoSRL learns faster with smaller networks

Figure 2 shows that ChronoSRL learns much faster than all baselines, with the highest AUC on every task and at every depth, except for a few humanoid settings with one or two layers in which no method learns. With eight or more layers, it needs a median of only 30 million steps, less than a third of the budget, to reach the final performance of the strongest baseline (Figure 2, last panel). We chose to report the AUC because it highlights this speed, but ChronoSRL also has the highest mean final ToG in all but one of these settings, with slightly smaller margins (see Figure 7 in Appendix E). ChronoSRL also needs much smaller networks, as in all four ant mazes, a critic with only four layers learns faster than every baseline at any depth up to 64. With the same number of environment steps, such a run takes only about a fifth of the wall-clock time of a baseline with 64 layers on the same GPUs (see Appendix C.3 for the run times). In the humanoid tasks, ChronoSRL needs eight layers for this, and only Humanoid U-Maze still clearly benefits from 64 layers.

4.2 What makes ChronoSRL work

To see where these gains come from, we study the components of ChronoSRL (see Figure 3 and Appendix F). The geometric losses are not specific to our critic, as adding them to SRL speeds it up

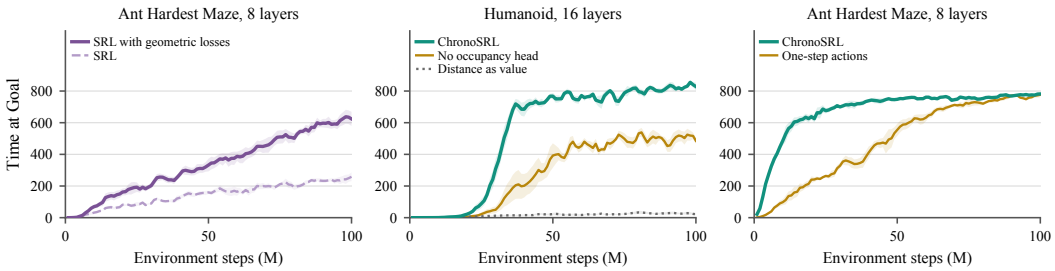


Figure 3: Added to SRL, the geometric losses more than double its AUC on Ant Hardest Maze, and removing the occupancy head or the action chunks slows down ChronoSRL. Without the occupancy head, the humanoid falls after reaching its goal, and a policy that follows the distance does not learn. All lines show the smoothed mean ToG over four seeds with one standard error.

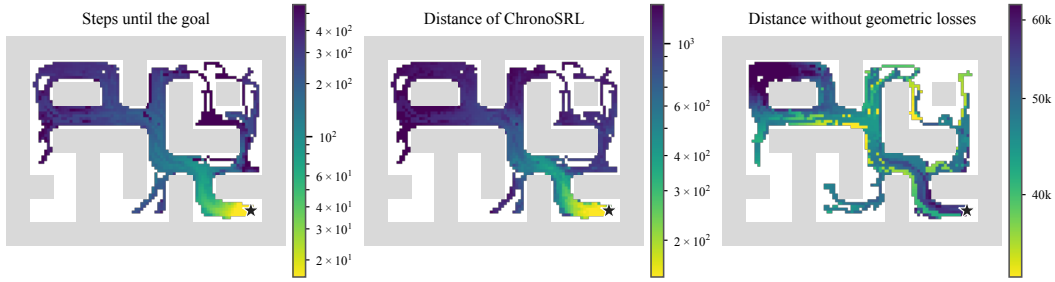


Figure 4: The distance of ChronoSRL follows the time that the agent needs to reach the goal. For a goal (star) in the far right corner of Ant Hardest Maze with 8 layers, the maps show the steps that the final policies still need from each position, the critic's distance to the goal converted to steps, and the same distance without the geometric losses, pooled over four seeds, each on its own color scale.

in all four ant mazes and more than doubles its AUC on Ant Hardest Maze. Within ChronoSRL, they mainly speed up early learning on the hardest tasks. The occupancy head is the major component that keeps the agent near its goal. Without it, the humanoid often falls after reaching the goal, as it overshoots with too much momentum, and loses about 40% of its final ToG. A policy that simply follows the distance instead of the predicted value does not learn at all, as the distance tells the agent neither how reliably it reaches the goal nor whether it stays there. Finally, the action chunks significantly speed up the start of learning process, as with single actions, ChronoSRL needs about three times as many steps to reach 80% of its final ToG on Ant U4-Maze and Ant Hardest Maze.

4.3 Inside the temporal geometry

In mazes, the distance between two states in space is a poor proxy to the actual time that reaching a goal takes. To see whether the critic of ChronoSRL measures the goal-reaching time properly, we start the ant at every free cell of a maze, let the final policy walk to the same goal, and, along the episodes that reach it, convert the critic's distance to the goal into steps by dividing it by κ (see Figure 4). In Ant Hardest Maze, the distance of ChronoSRL follows the steps that the agent actually needs, with a strong rank correlation of 0.93, while without the geometric losses, the distances are at tens of thousands of steps and, across the maze, are not even clearly positively related to the remaining steps. The difference between the distances in space and time is the largest in Ant U4-Maze, where the goal lies behind a wall, so that positions close to it in space can be far from it in time, like the state 3 in Figure 1. There, the distance of ChronoSRL follows the steps that the agent still needs, while without the geometric losses, it partly follows the straight-line distance instead (see Appendix G for more details). Both geometric losses are needed for this, as without the time loss, the distances are unrelated to the remaining steps, and without the separation loss, they follow the straight-line distance instead. The separation loss also stretches the scale of the distances, which can lead to overestimations of the remaining steps (see Appendix G for the maps of all variants). For control, this overestimate does not matter, as the policy does not act on the distance but on the

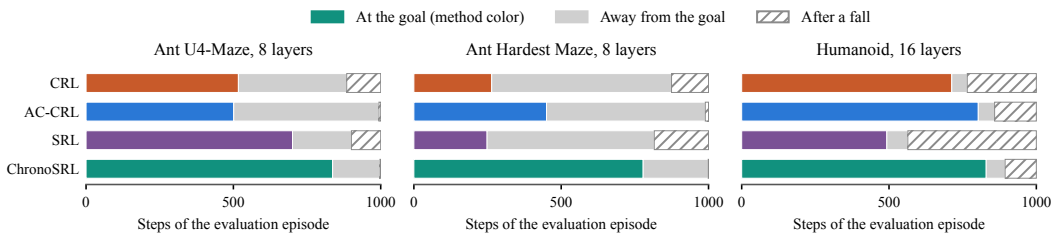


Figure 5: ChronoSRL spends most of each evaluation episode at the goal. Bars split the 1000 steps into the time at the goal (colored), away from it (gray), and after a fall (hatched) for the final policies.

predicted hazard and occupancy values, which matches the discounted waiting time that the agent actually experiences along evaluation episodes, with a rank correlation of 0.99 in both ant mazes.

4.4 Getting to the goal and staying there

To see how ChronoSRL achieves its higher ToG, we split each evaluation episode into the time at the goal, the time away from it, and the time lost after a fall. Figure 5 shows that in all four ant mazes, ChronoSRL with eight layers reaches the goal in more than 97% of the evaluation episodes, against 34 to 92% for the baselines (see Table 2 in Appendix E for more details), and then stays there, while the baselines spend far more time away from the goal, and SRL and CRL regularly flip over and fall. In the Humanoid environment, every method reaches the goal in more than 90% of the episodes, but afterwards SRL loses more than 40% of the episode because the humanoid falls, while ChronoSRL keeps the humanoid upright near the goal, which is exactly what our occupancy head encourages.

4.5 Toward self-supervised robot locomotion

Beyond the standard benchmarks, we evaluate self-supervised RL on more realistic robotic tasks and study whether these methods can learn the behaviors that legged robots need for real world deployment, with three tasks for the Unitree Go2 quadruped, implemented in the RL-X framework (Bohlinger & Dorer, 2023), including the necessary domain randomization, shaping terms, and learning curricula. In velocity tracking, the goal is a command for the forward, lateral, and yaw velocity of the robot's base, in goal-position reaching, it is a position of up to five meters away together with a heading angle, and in box climbing, the goal is on top and in the center of a box whose height grows with a performance-based curriculum (Magnus et al., 2025). All methods add their actions to a trot-shaped gait prior, which we found necessary to speed up learning, while the rest of the action and observation space follow the standard pipeline in legged robot locomotion (Rudin et al., 2022). We scale up training to 4096 parallel environments for 500 million steps, and both survival critics, SRL and ChronoSRL, receive the same shaping terms of the pipeline as additional time (as described in Section 3.4), which the contrastive critics cannot use.

Figure 6 shows that, from four layers on, ChronoSRL is the only method that learns to stay at its goal for most of the episode in velocity tracking and goal-position reaching, and at the end of training, it tracks the forward command with an error of about 0.1 to 0.2 meters per second, against 0.3 to 0.6 for the baselines (more details in Appendix H). In velocity tracking, the episodes of CRL and AC-CRL end within about 40 steps, as they just learn to fall into the direction of the commanded velocity. On box climbing, from eight layers on, ChronoSRL gets to the highest boxes during training, and its curriculum is still rising, while those of CRL and AC-CRL peak early and then get worse or stagnate. These results are a first step toward self-supervised RL for real robotics tasks, but they also show where the current limitations are. Without an additional cost for walking forward, ChronoSRL learns to reach goal positions by walking backward, and since we found that this cost keeps the box-climbing policies below a few centimeters, we remove it for that task and the policies climb the box backward. The tracking error is also not yet at the level of a reward-based PPO (Schulman et al., 2017), which reaches a tracking error of only 0.07 meters per second, and in further tests with the Unitree G1 humanoid, the policies did not learn to walk stably at all. Also the trot-shaped gait prior, while not harming sim-to-real transfer, was necessary to get the learning going in the first place. These are all issues that current reward-based methods overcome easily

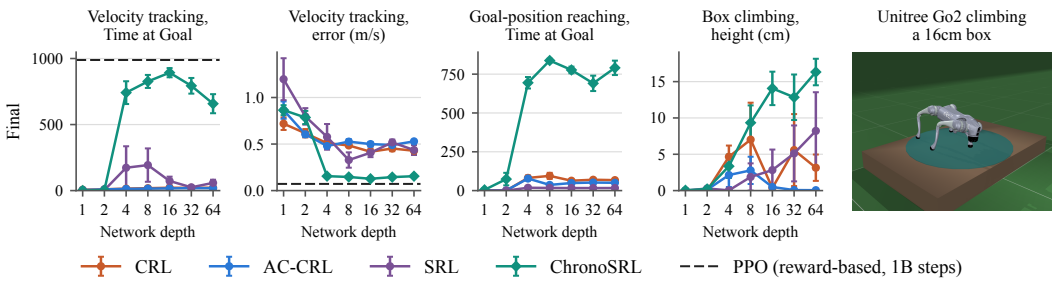


Figure 6: ChronoSRL learns velocity tracking and goal-position reaching with the Go2 quadruped and, from eight layers on, ends training with the highest box. The panels show the final ToG out of 1000 steps, tracking error, and curriculum box height as mean over four seeds with one standard error. We add a PPO baseline trained for 1B steps on the velocity tracking task for comparison. The image shows a 64-layer ChronoSRL policy climbing a 16cm box, with the goal-region in cyan.

in practice, which opens up clear directions for future research on contrastive and survival-based self-supervised RL methods.

5 Discussion and Limitations

Since the goal-reaching times come from the agent's own behavior, the geometry of ChronoSRL describes what the agent can currently do rather than the shortest paths that are possible. This makes it a natural basis for planning with subgoals that the agent can actually reach, which we leave for future work. Among the components of ChronoSRL, the geometric losses speed up learning most on the hardest tasks, and the survival critic that we build around them, with our occupancy head and action chunks, accounts for most of the gain over SRL. Predicting how long the agent stays at the goal, and not only when it arrives, proved important on Humanoid and the quadruped, and we expect it to matter for many practical tasks that reward staying rather than arriving, such as holding a pose or a velocity.

Our temporal supervision relies on trajectories that make progress toward their relabeled goals. Such progress is rare in the arm manipulation tasks of JaxGCRL, where the goal is the position of an object that rarely moves, so both survival critics, ChronoSRL and SRL, fail there, unlike the contrastive critics (see Appendix I for the arm results). Here, goals that also contain the position of the gripper could make progress visible, since the gripper moves toward the object long before the object itself moves. The same temporal geometry could also guide exploration by proposing goals at the edge of what the agent can currently reach (Pitis et al., 2020). Beyond the observation window, the critic only knows that a goal is far away, so tasks with longer horizons may benefit or even strictly require bootstrapping goal-reaching times beyond W steps.

6 Conclusion

We introduced ChronoSRL, a self-supervised RL method whose critic represents goal-reaching time as distance. It trains these distances to match the times that every trajectory provides for free, and it predicts from the same embeddings when the goal will be reached and how long the agent will stay there. On seven locomotion and navigation benchmarks, ChronoSRL typically reaches the final performance of contrastive, action-chunked contrastive, and survival baselines in less than a third of their training budget, and even with much smaller networks, it learns faster than baselines with up to 64 layers. On realistic and challenging quadruped locomotion tasks, it is the only one of these methods that learns to stay at commanded velocities and goal positions, and climbs the highest boxes. ChronoSRL shows that the time at which a goal is reached is not only what an agent should minimize, but also a signal that should directly shape the representation it learns.

Reproducibility statement

Section 3 and Appendix B describe the full method, Algorithm 1 summarizes the training procedure, and Table 1 lists all hyperparameters. Appendix C specifies the network architectures, the training setup, and the configurations of the baselines, and Appendix D defines the evaluation protocol, including seeds and aggregation. Appendix H specifies the robot tasks and their costs, and Appendix A contains the derivations behind our temporal targets. All results use four seeds per setting, and we will release the code for all methods, environments, and experiments upon acceptance.

Use of large language models

We used large language models, including OpenAI Codex and Anthropic's Claude, as research and writing assistants. They helped develop and critique ideas, start experiments, process and analyze experiment logs and draft the figures. The authors reviewed all content and take full responsibility for the paper, including all material produced with the help of language models.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Junik Bae, Kwanyoung Park, and Youngwoon Lee. Tldr: Unsupervised goal-conditioned rl via temporal distance-aware representations. In *Conference on Robot Learning*, 2024.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado Van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Nico Bohlinger. Intrinsically motivated agents for goal discovery in high dimensional state spaces. Master's thesis, Goethe University Frankfurt am Main, 2023. URL `https://publikationen.ub.uni-frankfurt.de/opus4/frontdoor/deliver/index/docId/75493/file/thesis_Bohlinger.pdf`.

Nico Bohlinger and Klaus Dorer. Rl-x: A deep reinforcement learning library (not only) for robocup. In *Robot World Cup*, pp. 228–239. Springer, 2023.

Nico Bohlinger, Grzegorz Czechmanowski, Maciej Krupka, Piotr Kicki, Krzysztof Walas, Jan Peters, and Davide Tateo. One policy to run them all: an end-to-end learning approach to multi-embodiment locomotion. In *Conference on Robot Learning*, 2024.

Magnus Victor Boock, Abdullah Akgül, Mustafa Mert Çelikok, and Melih Kandemir. Hitting time isomorphism for multi-stage planning with foundation policies. *arXiv preprint arXiv:2605.06470*, 2026.

Michał Bortkiewicz, Władysław Pałucki, Vivek Myers, Tadeusz Dziarmaga, Tomasz Arczewski, Łukasz Kuciński, and Benjamin Eysenbach. Accelerating goal-conditioned reinforcement learning algorithms and research. In *International Conference on Learning Representations*, volume 2025, pp. 15732–15754, 2025.

David R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B*, 34(2):187–202, 1972.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations*, 2021.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

Carlos Florensa, Jonas Degrave, Nicolas Heess, Jost Tobias Springenberg, and Martin Riedmiller. Self-supervised learning of image embedding for continuous control. *arXiv preprint arXiv:1901.00943*, 2019.

Michael F Gensheimer and Balasubramanian Narasimhan. A scalable discrete-time survival model for neural networks. *PeerJ*, 7:e6257, 2019.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations*, 2021.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 297–304. JMLR Workshop and Conference Proceedings, 2010.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pp. 1735–1742. IEEE, 2006.

Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. In *International Conference on Learning Representations*, 2020.

Leslie Pack Kaelbling. Learning to achieve goals. In *Ijcai*, volume 2, pp. 1094–8, 1993.

Edward L Kaplan and Paul Meier. Nonparametric estimation from incomplete observations. *Journal of the American statistical association*, 53(282):457–481, 1958.

Jared L Katzman, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC medical research methodology*, 18(1):24, 2018.

Michal Korniak, Kamil Dybek, Benjamin Eysenbach, Marco Bagatella, and MichaŁ Bortkiewicz. Three steps at a time: Learning representations from action sequences in contrastive rl. *arXiv preprint arXiv:2608.30640*, 2026.

Håvard Kvamme, Ørnulf Borgan, and Ida Scheel. Time-to-event prediction with neural networks and cox regression. *Journal of machine learning research*, 20(129):1–30, 2019.

Changhee Lee, William Zame, Jinsung Yoon, and Mihaela Van Der Schaar. Deephit: A deep learning approach to survival analysis with competing risks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Hojoon Lee, Dongyoon Hwang, Donghu Kim, Hyunseung Kim, Jun Jet Tai, Kaushik Subramanian, Peter Wurman, Jaegul Choo, Peter Stone, and Takuma Seno. Simba: Simplicity bias for scaling up parameters in deep reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 46816–46848, 2025.

Qiyang Li, Zhiyuan Paul Zhou, and Sergey Levine. Reinforcement learning with action chunking. *Advances in Neural Information Processing Systems*, 38:55518–55553, 2026.

Bo Liu, Yihao Feng, Qiang Liu, and Peter Stone. Metric residual network for sample efficient goal-conditioned reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 8799–8806, 2023.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*, 2022.

Leon Magnus, Oleg Arenz, and Nico Bohlinger. Investigating the influence of curriculum learning on quadrupedal parkour. 2025.

Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. Discovering and achieving goals via world models. *Advances in neural information processing systems*, 34:24379–24391, 2021.

Vivek Myers, Chongyi Zheng, Anca Dragan, Sergey Levine, and Benjamin Eysenbach. Learning temporal distances: Contrastive successor features can provide a metric structure for decision-making. In *International Conference on Machine Learning*, 2024.

Vivek Myers, Bill Zheng, Benjamin Eysenbach, and Sergey Levine. Offline goal-conditioned reinforcement learning with quasimetric representations. *Advances in Neural Information Processing Systems*, 38:19654–19679, 2026.

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample efficient continuous control. *Advances in neural information processing systems*, 37:113038–113071, 2024.

Franki Nguimatsia-Tiofack, Fabian Schramm, Théotime Le Hellard, and Justin Carpentier. Survival reinforcement learning: Toward scalable self-supervised rl. *arXiv preprint arXiv:2605.31273*, 2026.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Daniel Palenicek, Florian Vogt, Joe Watson, Ingmar Posner, and Jan Peters. Xqc: Well-conditioned optimization accelerates deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36:34866–34891, 2023.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. In *International Conference on Machine Learning*, 2024a.

Seohong Park, Oleh Rybkin, and Sergey Levine. Metra: Scalable unsupervised rl with metric-aware abstraction. In *International Conference on Learning Representations*, volume 2024, pp. 18579–18603, 2024b.

Silviu Pitis, Harris Chan, Stephen Zhao, Bradly Stadie, and Jimmy Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International conference on machine learning*, pp. 7750–7761. PMLR, 2020.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on robot learning*, pp. 91–100. PMLR, 2022.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320. PMLR, 2015.

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

Franki Nguimatsia Tiofack, Fabian Schramm, Théotime Le Hellard, and Justin Carpentier. Svl: Goal-conditioned reinforcement learning as survival learning. *arXiv preprint arXiv:2604.17551*, 2026.

Srinivas Venkattaramanujam, Eric Crawford, Thang Doan, and Doina Precup. Self-supervised learning of distance functions for goal-conditioned reinforcement learning. *arXiv preprint arXiv:1907.02998*, 2019.

Kevin Wang, Ishaan Javali, Michał Bortkiewicz, Tomasz Trzcinski, and Benjamin Eysenbach. 1000 layer networks for self-supervised rl: Scaling depth can enable new goal-reaching capabilities. *Advances in Neural Information Processing Systems*, 38:157643–157670, 2026.

Tongzhou Wang and Phillip Isola. On the learning and learnability of quasimetrics. In *International Conference on Learning Representations*, 2022.

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning*, pp. 36411–36430. PMLR, 2023.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023.

Bill Chunyuan Zheng, Vivek Myers, Benjamin Eysenbach, and Sergey Levine. Multistep quasimetric learning for scalable goal-conditioned reinforcement learning. In *International Conference on Learning Representations*, 2026.

Chongyi Zheng, Benjamin Eysenbach, Homer Walke, Patrick Yin, Kuan Fang, Ruslan Salakhutdinov, and Sergey Levine. Stabilizing contrastive rl: Techniques for robotic goal reaching from offline data. In *International Conference on Learning Representations*, volume 2024, pp. 28236–28264, 2024a.

Chongyi Zheng, Ruslan Salakhutdinov, and Benjamin Eysenbach. Contrastive difference predictive coding. In *International Conference on Learning Representations*, volume 2024, pp. 47577–47601, 2024b.

A TEMPORAL DISTANCE AND ACTION VALUE

Here, we explain why ChronoSRL does not use its temporal distances directly as action values. Throughout, we fix a state, an action chunk, a goal, and the policy that acts after the chunk, and we omit these fixed arguments by writing $T \in \{0, 1, \ldots\} \cup \{\infty\}$ for the first goal-reaching time.

We start from the hazard value of Equation 1, which counts the discounted steps spent waiting for the goal within the window. For every T, this number is

$$\sum_{t=0}^{W-1} \gamma^t \, \mathbf{1}\{T > t\} = \frac{1 - \gamma^{\min(T,W)}}{1 - \gamma}. \tag{6}$$

Taking the expectation under the true survival function $S(t) = \Pr(T > t)$ then turns the hazard value into $Q_{\text{hazard}} = -(1 - V)/(1 - \gamma)$ with $V = \mathbb{E}[\gamma^{\min(T,W)}]$. This is the finite-window form of the value identity of Tiofack et al. (2026, Proposition 4.1) used by SRL, and because the sum is finite, the identity also holds when the goal may never be reached. Since the value thus increases with V, the following proposition takes a closer look at V and splits it into the probability of reaching the goal within the window and the time it takes when the goal is reached.

Proposition 1 (Probability and time of goal reaching)**.** *Let* $0 < \gamma < 1$*,* $\kappa = -\log \gamma$*,* $p = \Pr(T \leq W) > 0$*, and* $\mu = \mathbb{E}[T \mid T \leq W]$*. Then*

$$V = p \, \mathbb{E}\big[e^{-\kappa T} \mid T \leq W\big] + (1 - p)\, e^{-\kappa W} \;\geq\; p\, e^{-\kappa \mu} + (1 - p)\, e^{-\kappa W}, \tag{7}$$

with equality if and only if T is almost surely constant given $T \leq W$.

Proof. Since $\min(T, W) = T$ on the event $T \leq W$ and $\min(T, W) = W$ on the event $T > W$, splitting the expectation over these two events gives the equality. For the inequality, note that the function $t \mapsto e^{-\kappa t}$ is strictly convex because $\kappa > 0$. Jensen's inequality therefore gives $\mathbb{E}[e^{-\kappa T} \mid T \leq W] \geq e^{-\kappa \mu}$, with equality if and only if T is almost surely constant given $T \leq W$, and multiplying by $p > 0$ yields the bound together with its equality condition. □

The time loss of Equation 2, in contrast, captures only the second of these two quantities, because it uses only the examples in which the goal was reached. For a fixed input and an unrestricted prediction, its minimizer therefore depends on the goal-reaching times of these examples but not on the probability p of reaching the goal. If, in addition, all residuals lie within one discount horizon, where the Huber loss is quadratic, this minimizer is simply the mean of κT over the reached examples.

To make this difference between the distance and the value concrete, we consider two actions that reach the goal after exactly 200 steps whenever they reach it, where the first action reaches the goal in 90% of the attempts and the second in only 10%. Both actions thus have the same target distance $\kappa \cdot 200 \approx 0.2$, yet their values differ. At $\gamma = 0.999$ and $W = 1000$, Equation 7 holds with equality and gives $V \approx 0.77$ for the first action and $V \approx 0.41$ for the second, which correspond to $Q_{\text{hazard}} \approx -226$ and $Q_{\text{hazard}} \approx -587$ discounted steps of waiting. A value computed from the distance alone, such as e^{-d}, nevertheless ranks both actions equally. Even for certain arrivals, such a value recovers V only when the goal-reaching time does not vary, since the inequality in Equation 7 is strict otherwise. The hazard head relies on neither assumption. Instead, it models the distribution of T within the window, and its likelihood also learns from the censored examples, which carry the information about p.

B LABELS, LOSSES, AND PREDICTION HEADS

Algorithm 1 summarizes the training of ChronoSRL, and the subsections below describe the labels, losses, and prediction heads it relies on.

B.1 RELABELING AND GOAL-REACHING TIMES

A window holds $W = 1000$ consecutive transitions, so it can contain the end of one episode and the beginning of the next. Within such a window, the relabeler follows SRL and gives each step i a future goal with probability 0.85, the current achieved goal with probability 0.05, and the achieved

Algorithm 1 ChronoSRL

```
Initialize the critic, the policy π, the entropy coefficient α, and the replay buffer
for each training step do
    Collect 62 steps in every parallel environment with π, executing chunks of two actions, and store the transitions
    Sample one window of W consecutive transitions per environment from the replay buffer
    Relabel goals and compute the labels of every example (see Appendix B.1)
    for each minibatch of relabeled examples do
        Update the critic on L_hazard + L_occ + L_time + L_sep (Section 3.3)
        Compute β on the minibatch and update π on J(π) (Equation 5), then update α
    end for
end for
```

goal of a random step of the window with probability 0.10. Future goals come from the later steps j of the same episode with probability proportional to γ_g^{j-i}, where $\gamma_g = 0.99$ is the goal discount of CRL, and the current goal is used whenever the episode has no later step. Once a goal is chosen, the goal-reaching time τ_i is the first offset after step i at which the episode enters $\mathcal{B}_\varepsilon(g_i)$, with $\tau_i = 0$ if s_i already lies in the goal region. The indicator δ_i is one if this offset lies within the window, and otherwise the example is censored at the full window W, as in SRL.

Following SRL, the humanoid tasks replace the point goal by a sequence of $k = 32$ consecutive achieved goals from the sampled step onward, and a random goal is repeated k times. A sequence is matched at a start step if the k achieved goals from there lie within ε of the corresponding elements, all within the same episode, and the time τ_i is then the earliest matching start relative to step i. The policy, however, observes only a point goal g and queries the critic with the repeated sequence $(g, \dots, g)$. Finally, an example is valid only if its chunk of two actions from step i onward lies within the window and the episode.

B.2 Time and separation losses on a batch

Both geometric losses are computed on a batch, where $D_{ij} = d_\theta(s_i, a_i, g_j)$ pairs the state and action of example i with the goal of example j and v_i indicates a valid example. The time loss then averages over the reached examples $\mathcal{R} = \{i \mid v_i = 1,\ \delta_i = 1,\ \tau_i \geq 1\}$,

$$\mathcal{L}_{\text{time}} = \frac{1}{\max(1, |\mathcal{R}|)} \sum_{i \in \mathcal{R}} \rho\big(D_{ii} - \kappa\tau_i\big), \qquad \rho(r) = \begin{cases} \frac{1}{2}r^2, & |r| \leq 1, \\ |r| - \frac{1}{2}, & |r| > 1. \end{cases} \tag{8}$$

The separation loss, in contrast, penalizes distances below the margin κW and places the censored examples $\mathcal{C} = \{i \mid v_i = 1,\ \delta_i = 0\}$ and the cross-batch pairs $\mathcal{M} = \{(i, j) \mid i \neq j,\ \Delta(g_i, g_j) > \varepsilon\}$ under one denominator,

$$\mathcal{L}_{\text{sep}} = \frac{\sum_{i \in \mathcal{C}} [\kappa W - D_{ii}]_+ + \sum_{(i,j) \in \mathcal{M}} [\kappa W - D_{ij}]_+}{\max\big(1, |\mathcal{C}| + |\mathcal{M}|\big)}. \tag{9}$$

Here the goal difference Δ is the Euclidean distance between point goals and, for goal sequences, the root mean square of the element-wise distances. Because the cross-batch pairs are selected by their goal difference alone, they may contain reachable goals, which is why they never enter the hazard likelihood. In both losses, distances are computed as $\big(\|\phi_\theta(s, a) - \psi_\theta(g)\|_2^2 + 10^{-12}\big)^{1/2}$.

B.3 Hazard head

We use the hazard head of SRL without changes (Nguimatsia-Tiofack et al., 2026). It predicts the probability that the goal is already reached together with one hazard for every step of the window, and with one bin per step the piecewise-constant survival parameterization of Tiofack et al. (2026) is exact. The likelihood then treats the two kinds of examples differently. A reached example contributes the log-probability of reaching the goal at step τ, which for $\tau = 0$ is the probability that the goal is already reached, whereas a censored example contributes the log-probability of not reaching the goal within the window. The loss $\mathcal{L}_{\text{hazard}}$ averages the negative log-likelihood over the valid examples, and the value of Equation 1 likewise adds no cost for the time after W.

B.4 Occupancy targets and value

The occupancy head is one additional linear output layer on the prediction network of the hazard head (see Appendix C.1). Its targets record how close the trajectory stays to the goal over time. For the goal g_i of example i, or the first element of a goal sequence, the target at an offset $t \geq 1$ averages the membership in three nested goal regions,

$$o_{it} = \frac{1}{3} \sum_{r \in \{\varepsilon,\, \varepsilon/2,\, \varepsilon/4\}} \mathbf{1}\big\{ s_{i+t} \in \mathcal{B}_r(g_i) \big\}. \tag{10}$$

Whether a target is observed depends on how the episode of step i ends. The target is observed while this episode continues within the window, and after a termination all later offsets of the window are observed with $o_{it} = 0$. Offsets after a truncation or beyond the window, by contrast, are unobserved. Rather than predicting every offset, the head groups the offsets into $L = 30$ bins with logarithmically spaced integer boundaries $0 = b_0 < b_1 < \cdots < b_L = W$, and the bin target $y_{i\ell}$ is the mean of the observed o_{it} in bin ℓ. Bins without an observation are masked. With the predicted logit $z_{i\ell}$ and the mask $m_{i\ell}$ of valid examples and observed bins, the occupancy loss is a binary cross-entropy that weighs every observed bin equally, independently of its width,

$$\mathcal{L}_{\mathrm{occ}} = \frac{\sum_{i,\ell} m_{i\ell} \big[\operatorname{softplus}(z_{i\ell}) - y_{i\ell}\, z_{i\ell} \big]}{\max\big(1, \sum_{i,\ell} m_{i\ell}\big)}. \tag{11}$$

To obtain the occupancy value of Equation 4, we hold the prediction constant within each bin and leave out the first bin, which only contains the offset zero,

$$Q_{\mathrm{occ}} = \sum_{\ell=1}^{L-1} \sigma(z_\ell) \sum_{t=b_\ell}^{b_{\ell+1}-1} \gamma^t, \tag{12}$$

where σ is the logistic function.

B.5 Occupancy weight and policy update

In the policy objective, the occupancy value is scaled by the weight β, the fraction of valid examples in the batch whose episode terminates within the window at or after their step,

$$\beta = \frac{\sum_i v_i e_i}{\max\big(1, \sum_i v_i\big)}, \tag{13}$$

where e_i indicates such a termination. This weight is computed without gradient and held fixed during the policy update. It only scales Q_{occ} in $J(\pi)$, whereas the occupancy head itself is trained in every update. The policy is a tanh-squashed diagonal Gaussian over all components of a chunk, and its entropy coefficient is tuned toward a target entropy of -0.5 per action dimension of the chunk (Haarnoja et al., 2018). When the policy is updated, the gradient with respect to the sampled chunk passes through both encoders and both heads of the fixed critic.

B.6 Shaping costs

Shaping replaces the plain step count by a clock that also accumulates costs. For a trajectory with per-step costs $c_t \geq 0$, the shaped clock starts at $C_0 = 0$ and advances as $C_{t+1} = C_t + 1 + \lambda c_t$, and the shaped time $\widetilde{\tau}$ from step i to a goal reached at step j is $C_j - C_i$, rounded to the nearest integer. A goal then counts as reached only if its rounded shaped time is at most W. With $\lambda > 0$, an example without such an arrival is censored at the rounded shaped time until the end of its window, clipped to W, while the separation margin remains κW. Occupancy observations are likewise assigned to bins by their rounded shaped time, and those beyond the last boundary are clipped into the last bin. They are neither interpolated nor weighted by the duration of their cost increment. The costs can take two forms. They can be the mean squared change between consecutive stored actions, which is zero for the first action of an episode, or the nonnegative weighted sum of the shaping penalties that the robot environment reports. Shaped time is implemented in the relabeling of goal sequences, which the robot tasks use, and with $\lambda = 0$ all labels, including the censoring at W, are exactly the unshaped labels (see Appendix B.1).

C Architecture and training

C.1 Networks

All networks follow the residual architecture of scaled CRL and SRL (Wang et al., 2026; Nguimatsia-Tiofack et al., 2026), in which each residual block contains four dense layers with layer normalization and Swish activations. As in Wang et al. (2026), the reported depth counts the dense layers in the residual blocks of one network. Because a residual block needs four layers, networks with one or two layers are plain stacks of such layers instead.

The critic first embeds its inputs with two encoders. The state–action encoder ϕ_θ receives the state and the flattened action chunk, whereas the goal encoder ψ_θ receives the goal or the flattened goal sequence. Both encoders have the reported depth and end in a linear projection to 64 dimensions without layer normalization. Their embeddings then enter the prediction network, which is the hazard network of SRL (Nguimatsia-Tiofack et al., 2026). This network modulates the state–action embedding with the goal embedding, processes both with a residual network of the reported depth, and outputs the instant probability and the hazards through a learned temporal basis of rank 64. For the occupancy head, we add a linear layer with 30 outputs to this network, so that both heads share all parameters except their output layers. The actor, finally, is a separate residual network of the reported depth that maps the state and the point goal to a Gaussian over action chunks.

C.2 Training configuration

Table 1 lists all training settings of ChronoSRL next to those of the reference implementation of SRL. We set all loss weights to one and tune no setting per task.

Table 1: On the seven benchmarks, ChronoSRL differs from the reference implementation of SRL only in the settings of the upper block.

Setting	SRL	ChronoSRL
Critic loss	$\mathcal{L}_{\text{hazard}}$	$\mathcal{L}_{\text{hazard}} + \mathcal{L}_{\text{occ}} + \mathcal{L}_{\text{time}} + \mathcal{L}_{\text{sep}}$
Value used by the policy	Q_{hazard}	$Q_{\text{hazard}} + \beta\, Q_{\text{occ}}$
Action chunk	One action	Two actions, open-loop
Future-goal sampling discount	0.999	0.99
Embedding dimension	128, layer normalization	64, no normalization
Encoder depth	Half the reported depth	The reported depth
Occupancy bins and radii	None	30 bins, radii ε, $\varepsilon/2$, $\varepsilon/4$
Budget and episode length	100M environment steps, episodes of 1000 steps	
Seeds and depths	1000 to 1003, depths 1, 2, 4, 8, 16, 32, and 64	
Parallel environments and unroll length	512 environments, 62 steps	
Minibatch updates per training step	Up to 800	
Replay size per environment	1000 to 10000 transitions	
Learning rates	3×10^{-4} for actor, critic, and entropy coefficient	
Hidden width	256	
Discount and observation window	$\gamma = 0.999$ and $W = 1000$ steps	
Hazard bins	1000 bins of one step	
Goal radius	$\varepsilon = 0.5$	
Goal mixture	0.85 future, 0.05 current, 0.10 random	
Humanoid goals and batch size	Sequences of $k = 32$ goals, 1024 examples	
Ant goals and batch size	Single goals, 512 examples	

C.3 Reference methods

All baselines run in our training pipeline with the reference configuration of their papers. Within this pipeline, all methods perform up to 800 updates for every 512×62 environment steps on minibatches of 512 examples, or 1024 for the survival critics on the humanoid tasks. Both CRL and AC-CRL use the forward InfoNCE objective with a log-partition penalty of 0.1, the discount 0.99, and embeddings with 64 dimensions, and AC-CRL additionally executes chunks of three actions. For SRL, we use the configuration of Table 1. Since every baseline keeps the configuration of its

paper, the comparisons with the complete baselines also include differences in architecture and goal sampling. The encoders of SRL, for example, have only half the reported depth. The advantage of a four-layer ChronoSRL critic over baselines with 64 layers, however, exceeds this factor by far.

We simply chose a budget of 100 million steps to keep the compute time of all experiments manageable. It is smaller than the 200 to 400 million steps that Wang et al. (2026) use for Ant Hardest Maze and the humanoid mazes. This smaller budget explains the low absolute ToG of all methods on the humanoid mazes, and with longer training all methods, including ChronoSRL, would likely improve there. Since ChronoSRL needs fewer layers, it also reaches a given performance at a lower cost. A four-layer run of ChronoSRL in the ant mazes, for example, takes 1.8 hours on GH200 GPUs, against 8.1 to 9.4 hours for the baselines with 64 layers on the same GPUs.

D Evaluation protocol

We train all four methods on the seven tasks at depths from 1 to 64 with the seeds 1000 to 1003, and each run lasts 100 million environment steps. Every million steps, we evaluate the policy on 100 episodes of 1000 steps.

We summarize each run by its normalized AUC and its final value. The normalized AUC is the mean of evaluations 1 to 100 and equals the area under the piecewise-constant learning curve divided by the training budget, whereas the final value is the mean of evaluations 96 to 100. The learning speed in the last panel of Figure 2 instead compares ChronoSRL with the baselines. It is the number of environment steps after which the mean ToG of ChronoSRL over four seeds, smoothed over three evaluations, first reaches the mean final ToG of the strongest baseline at the same depth. Note that the highest possible ToG lies below 1000, since the agent first has to travel to the goal. In Ant U4-Maze, for example, where the final policies of ChronoSRL need a median of about 130 steps to reach the goal, it lies at about 870.

We compute each summary per seed and report the mean over the four seeds together with the standard deviation or the standard error over seeds. Tables give means and sample standard deviations, and plots give means and standard errors.

E Benchmark results

Figures 2 and 7 plot the normalized AUC and the final ToG against network depth (see Appendix D). Once the networks have eight or more layers, ChronoSRL has the highest normalized AUC on every task. Its mean final ToG is also the highest in every setting except Ant U5-Maze with 64 layers, where SRL is slightly ahead with 768 against 756 steps. Beyond these two metrics, Table 2 shows that, with eight layers, ChronoSRL reaches the goal in almost every episode of the four ant mazes.

Table 2: Each cell gives the percentage of evaluation episodes in which an eight-layer agent reaches the goal at least once, averaged over the last five evaluations and four seeds.

Task	CRL	AC-CRL	SRL	ChronoSRL
Ant U4-Maze	67	66	85	99
Ant Big Maze	81	92	88	98
Ant Hardest Maze	41	70	34	99
Ant U5-Maze	39	56	50	99
Humanoid	93	94	94	94

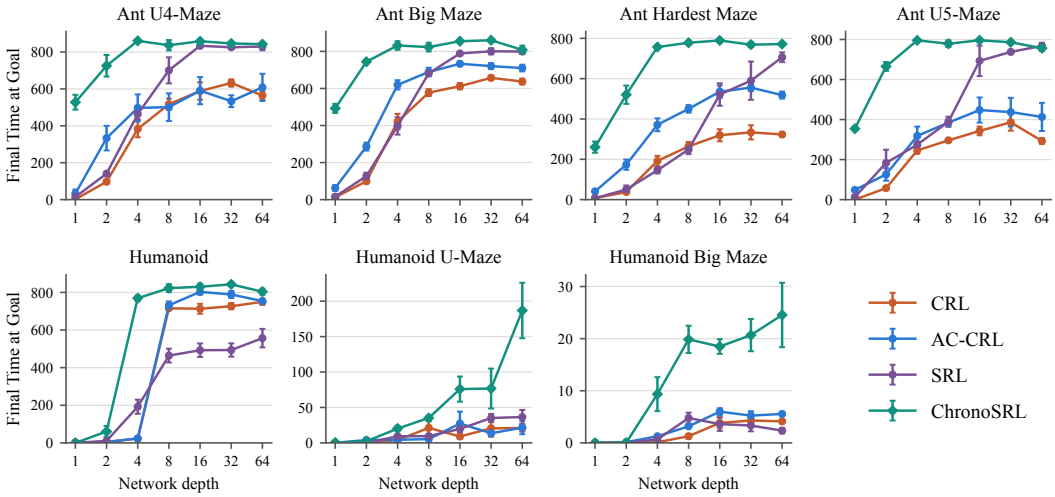


Figure 7: Points show the mean final ToG over four seeds with one standard error.

E.1 LEARNING CURVES AT EVERY DEPTH

Figure 8 complements these summaries with the learning curves of all methods at every depth.

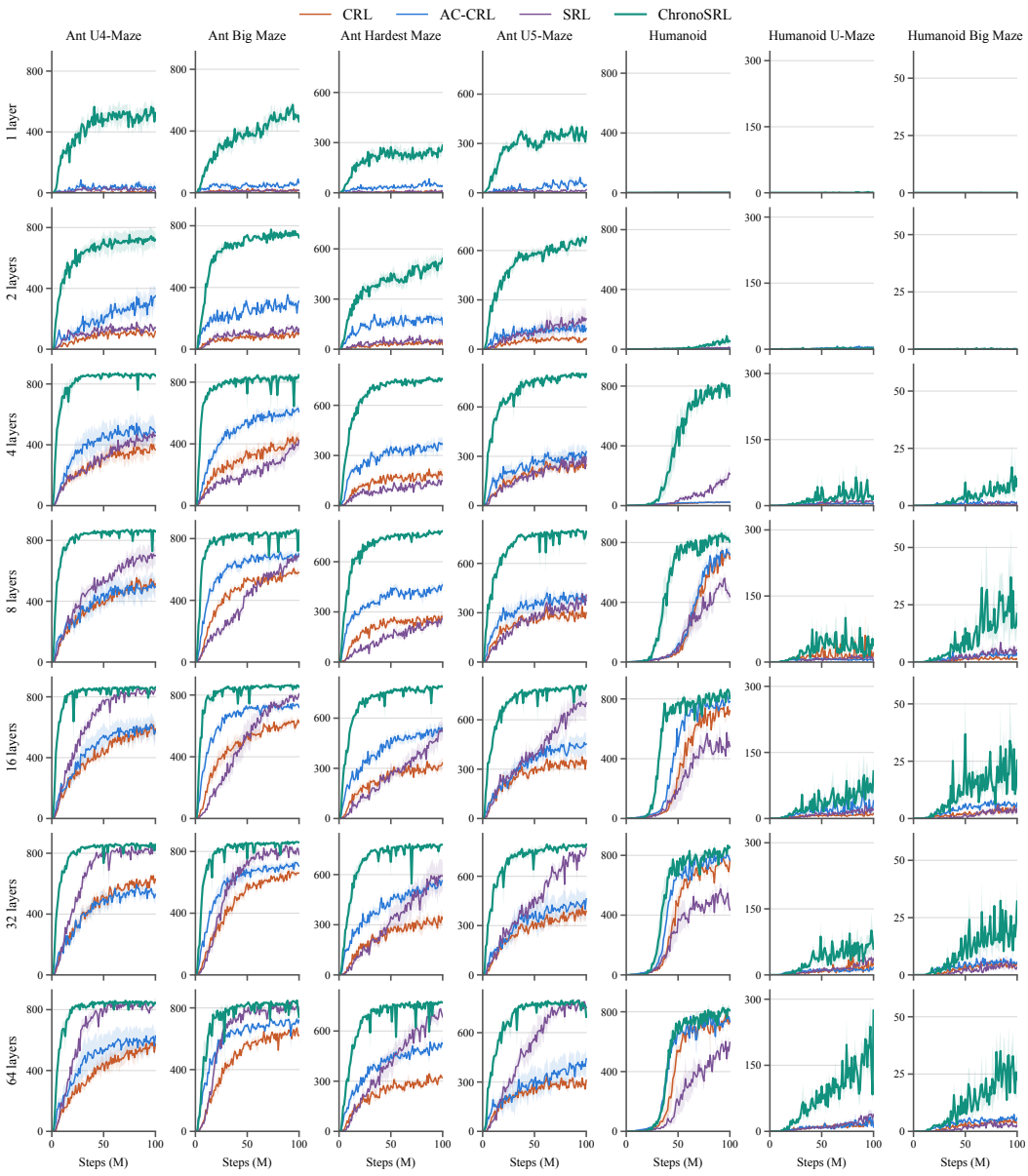


Figure 8: Rows show the network depth and columns the task, and each line shows the mean ToG over four seeds with one standard error.

F Component ablations

To see what individual components contribute, we ablate ChronoSRL on Humanoid with 16 layers and on Ant Hardest Maze and Ant U4-Maze with 8 layers. Each ablation removes one component and keeps all other settings and the seeds of ChronoSRL. The matched base is ChronoSRL without the time and separation losses, again with all other settings unchanged. Rather than removing the time loss, the variant with permuted goal-reaching times shuffles its targets among the reached examples of each batch, which keeps their distribution but removes their relation to the examples. The variant without the occupancy head drops it from both the critic loss and the value of the policy. The distance actor, in contrast, keeps all critic losses and maximizes $-d_\theta(s, a, g)/(1-\gamma)$ with the same goal query. Finally, the one-step variant executes single actions but still counts its labels in environment steps. As for their effects, removing the occupancy head lowers the final ToG on Humanoid from 830 to 509, because the humanoid falls after reaching its goal. Single actions, on the other hand, slow down learning, so that ChronoSRL reaches 80% of its final ToG only after 56 instead of 18 million steps on Ant Hardest Maze and after 29 instead of 9 million steps on Ant U4-Maze.

F.1 Geometric losses in another critic

To test whether the geometric losses also help another critic, we add them with weight one to SRL in its complete reference configuration and train it on the seeds of our SRL baseline. In particular, this configuration keeps the normalized embeddings, half-depth encoders, goal sampling, single actions, and value of SRL. Figure 9 shows that the losses speed up SRL in all four ant mazes with 8 layers, and in each of these mazes every seed with the losses has a higher AUC than every seed without them. On Ant Hardest Maze, the losses more than double the normalized AUC of SRL, from 143 to 335, and in the other three mazes SRL with the losses even reaches the final ToG of ChronoSRL. On Humanoid with 16 layers, however, three of the four seeds of SRL with the losses do not learn, which we attribute to a conflict between the losses and the normalized embeddings of SRL. Still, the survival critic of ChronoSRL alone, without the geometric losses, already improves on SRL more than the losses do on each of these tasks. On Ant Hardest Maze, for example, this matched base reaches a normalized AUC of 647.

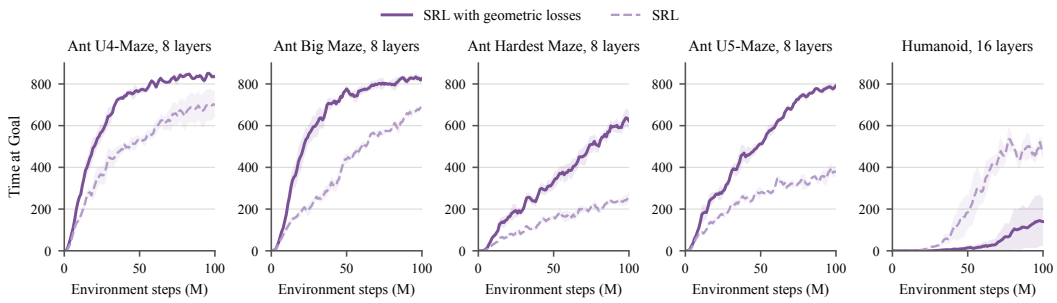


Figure 9: Added to SRL in its reference configuration, the geometric losses speed up learning in all four ant mazes, but three of the four seeds do not learn on Humanoid. Lines show the smoothed mean ToG over four seeds with one standard error.

F.2 Results of the ablations

Table 3 gives the normalized AUC of all ablations alongside that of ChronoSRL. Without either geometric loss, the AUC drops on Humanoid and Ant Hardest Maze, whereas on Ant U4-Maze both variants learn equally fast. Without the occupancy head, the humanoid still reaches the goal in 98% of the evaluation episodes, yet it loses 426 of the 1000 steps after a fall, against 106 for ChronoSRL. A policy that follows the distance, by contrast, hardly learns on any task. Finally, single actions lower the AUC on every task, most of all on Ant Hardest Maze.

Table 3: Following the distance fails on every task, while single actions slow down learning everywhere. On Humanoid, every ablation lowers the AUC, and removing the occupancy head nearly halves it. Entries show the normalized AUC as mean $\pm$ sample standard deviation over four seeds.

Variant	Humanoid, 16 layers	Ant Hardest Maze, 8 layers	Ant U4-Maze, 8 layers
ChronoSRL	541 ± 14	682 ± 16	801 ± 18
Neither geometric loss	508 ± 39	647 ± 42	800 ± 11
Permuted times	505 ± 45	674 ± 11	800 ± 21
No occupancy head	286 ± 22	681 ± 19	801 ± 6
One-step actions	475 ± 51	486 ± 33	710 ± 14
Distance as value	17 ± 7	70 ± 17	70 ± 15

G Measuring temporal geometry

Here, we describe how we measure the temporal geometry analyzed in Section 4.3. We run the final deterministic policies of all four seeds and record the distance to the goal at every decision step before the first arrival. After converting this distance to steps by dividing it by κ, we compare it with the steps that remain until the goal is first reached. Throughout, all correlations are Spearman rank correlations, and values with $\pm$ are means and standard errors over four seeds.

G.1 Distance maps

To build the distance maps, we place the ant at the center of every free cell of the maze, with the reset noise of the environment and a uniform offset of up to half a meter. In Ant Hardest Maze, the goal is the center of the far corner, and we run four episodes of 1000 steps from every cell. Ant U4-Maze instead has its goal at the center of the dead end, and there we run eight episodes of 500 steps from every cell. Each map then shows the median over all four seeds within squares of half a meter, taken along the episodes that reach the goal.

We measure how closely the distance follows time with rank correlations, for which each variant is compared with the remaining steps of its own final policy. For the map in Figure 4, the rank correlation between distance and remaining steps is 0.93 for ChronoSRL, whereas it is negative without the geometric losses. To separate time from space in Ant U4-Maze, we further compute per seed the partial rank correlation of the distance with the remaining steps after removing the ranks of the straight-line distance to the goal, as well as the converse. This shows that the distance of ChronoSRL follows the remaining steps, with a partial rank correlation of 0.62, and hardly depends on the straight-line distance. Without the geometric losses, by contrast, the distance depends on both about equally.

G.2 Roles of the two geometric losses

To separate the roles of the two geometric losses, we repeat this analysis for all variants on Ant Hardest Maze with four layers and on Ant U4-Maze with eight layers, and Figure 10 shows the resulting maps. The time loss is what orders the distances by time, since without it the distances in Ant Hardest Maze are unrelated to the remaining steps. The separation loss, in turn, matters only where time and space differ. In Ant Hardest Maze, the remaining steps and the straight-line distances are strongly related, whereas in Ant U4-Maze, where they differ, the distance without the separation loss follows the straight-line distance instead of the remaining steps. Only the two losses together thus make the distance follow the time of the agent. In addition, the separation loss stretches the scale, so that the distance of ChronoSRL overestimates the remaining steps several-fold in both mazes.

G.3 Hazard value along evaluation episodes

Besides the maps, we also examine the critic along evaluation episodes. We run 64 evaluation episodes of ChronoSRL and of its variant without the geometric losses from the standard start of Ant U4-Maze and Ant Hardest Maze, and at every decision step we query the critic with the chunk that the policy plans there. Figure 11 shows that along these episodes, the distance of ChronoSRL shrinks

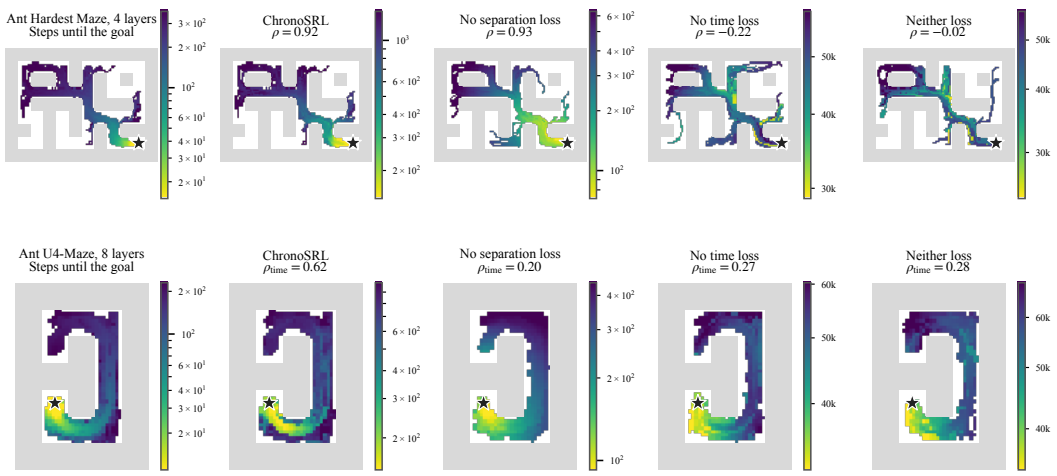


Figure 10: Both geometric losses are needed for the distance to follow the time of the agent. The maps show the steps until the goal for the final policies of ChronoSRL and the distance of each variant converted to steps, pooled over four seeds and each on its own color scale. The titles give the mean rank correlation with the remaining steps, which in Ant U4-Maze is a partial correlation that accounts for the straight-line distance.

from about one discount horizon at the start to about 150 steps, whereas without the geometric losses it stays at tens of thousands of steps.

The same queries also let us check the hazard value Q_{hazard} of the planned chunk, which predicts a discounted waiting time. We compare this prediction with the waiting time that the agent actually experiences, $(1-\gamma^{T-t})/(1-\gamma)$ for a first arrival at step T. In both ant mazes, the rank correlation between the two is 0.99. The prediction is nevertheless longer by a median of 11 to 16%, since the value is trained on the stochastic behavior in the replay rather than on the deterministic evaluation policy.

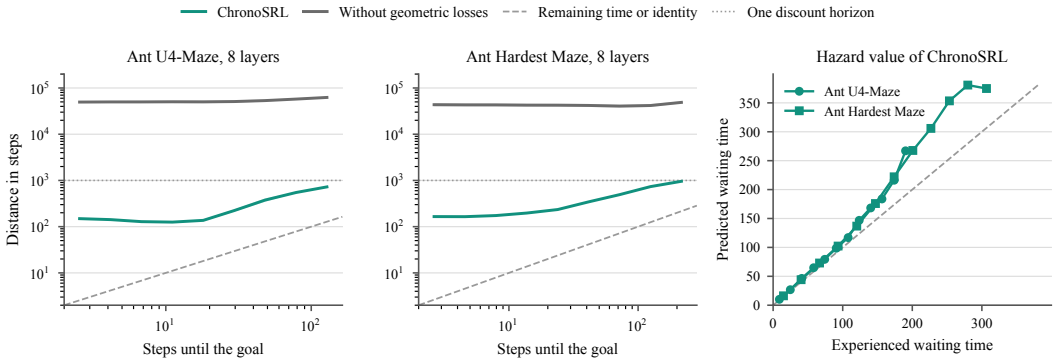


Figure 11: Along evaluation episodes, the distance of ChronoSRL shrinks with the steps until the goal, and its hazard value tracks the discounted waiting time that the agent experiences. The left panels show the distance in steps as the geometric mean over four seeds of the per-run medians within logarithmically spaced bins of the remaining steps, with one standard error. The right panel shows the median predicted waiting time within bins of the experienced one, with the identity as a dashed line.

H QUADRUPED ROBOT LOCOMOTION TASKS

H.1 SETUP AND TASKS

All robot experiments use the Unitree Go2 in the locomotion environment of RL-X (Bohlinger et al., 2024), which we simulate with MuJoCo-Warp in 4096 parallel environments at 50 Hz with episodes of 1000 steps. We keep the domain randomization, observation noise, shaping terms, and per-environment curriculum of this environment, and the curriculum coefficient c scales both the randomization and the shaping. The same coefficient enters the termination rule, as a training episode ends when the trunk drops below $(1-c)$ times 60% of its standing height. The actor observes the proprioception of RL-X, the phase of a gait clock, and the goal in the robot's frame from the true pose, but it receives no height map. The actions of all methods are added to a trot-shaped prior on the joint targets with a base amplitude of 0.3 and a frequency of 3 Hz, and this prior stays on during evaluation. Every run trains for 100 epochs of 5 million steps with batches of 8192 examples. Evaluation instead runs in 4096 separate parallel environments at curriculum level zero, where an episode ends only when the robot falls.

The three tasks differ only in their goal definition. In velocity tracking, the goal is the RL-X command for the forward, lateral, and yaw velocity of the base. Each component of this command is sampled uniformly from $[-1, 1]$ and set to zero below 0.1, and about 8% of the commands are standing commands. A goal counts as reached when the norm of the velocity error is below 0.25. We report the time at the goal, the tracking error per step, and the episode length on a fixed quarter of the evaluation environments, which are commanded to walk forward at 1 meter per second.

The other two tasks give the robot a goal pose instead. In goal-position reaching, this pose lies 1 to 5 meters away in any direction, and its heading points from the start toward the goal. We encode it as the position together with 0.3 times the cosine and sine of the heading, and the goal counts as reached within a distance of 0.5 in this encoding. Box climbing adds a box centered on the goal, whose height is 36 centimeters times the curriculum coefficient. For goal-position reaching we report the time at the goal, whereas for box climbing we report the box height of the training curriculum, since evaluation runs at curriculum level zero. The curriculum of each environment rises after an episode in which the trunk stood at least 20 centimeters above the box top, within its footprint, for 50 consecutive steps, and it falls after any other episode, with a step that grows with the number of consecutive successes or failures.

H.2 CRITICS AND SHAPING COSTS

ChronoSRL uses its benchmark configuration and SRL its reference configuration, both with goal sequences of $k = 8$. The contrastive baselines CRL and AC-CRL use a discount of 0.99, an embedding dimension of 64, and future goals only, with chunks of three actions for AC-CRL. The two survival critics of ChronoSRL and SRL receive the shaping terms of RL-X as the per-step cost c_t of the shaped time in Section 3.4, with $\lambda = 10$. Concretely, this cost is the negated sum of the 23 penalty terms of the RL-X reward with their weights and curriculum scaling, clipped to $[0, 50]$. In goal-position reaching and box climbing, the curriculum scale of this cost is at least 0.3.

Goal-position reaching adds one more term, a travel cost of $0.15\,(1 - \cos\vartheta)\min(v/0.2, 1)$ per step that both survival critics pay and that the curriculum does not scale. Here, ϑ is the angle between the horizontal velocity of the trunk and its heading, and v is the speed in meters per second. Without this cost, ChronoSRL reaches its goals by walking backward. With it, however, the box-climbing policies flip over at the edge of the box from 4 centimeters on, at every depth we tried. Box climbing therefore runs without the travel cost, and its policies climb the box mostly backward.

As a reference, we also train reward-based PPO (Schulman et al., 2017) on velocity tracking with the default hyperparameters of RL-X and an entropy coefficient of 0.002, for about 1 billion steps with four seeds. PPO uses the same environment, randomization, and curriculum, but it trains on the RL-X reward without the trot prior and resamples commands within episodes. Its final policies track the forward command with an error of 0.07 meters per second, and after 500 million steps, the budget of our runs, the error is already 0.085.

H.3 RESULTS

Figure 12 adds the episode length in velocity tracking to the final values of Figure 6. It shows that from four layers on, ChronoSRL tracks the forward command most accurately and stays at velocity and position goals longest. As a reference for the episode length, a robot that sends no actions to the gait prior falls after 54 steps, but the velocity-tracking episodes of CRL and AC-CRL end even earlier, after 32 to 41 steps.

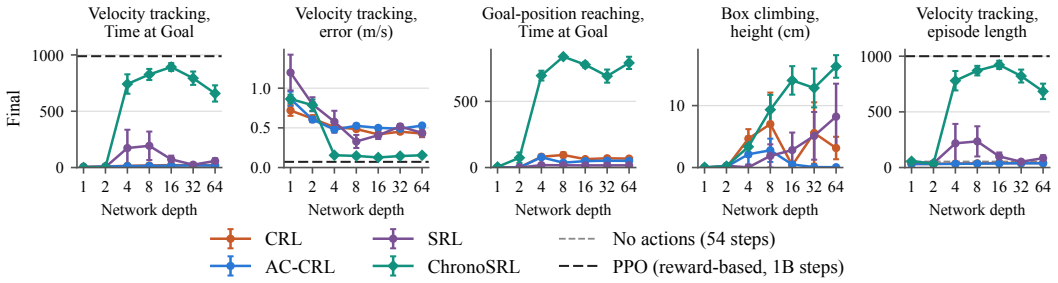


Figure 12: At the end of training and from four layers on, ChronoSRL stays at velocity and position goals longest and tracks the forward command most accurately, while the contrastive baselines end their velocity-tracking episodes even before the dashed line, at which a robot that sends no actions falls. Points show the mean of the last five evaluations over four seeds with one standard error.

The learning curves in Figure 13 show that the box curricula of CRL and AC-CRL rise fastest, with single runs reaching boxes of up to 25 centimeters early in training. From four layers on, however, these curricula later fall back at every depth. The curriculum of ChronoSRL rises more slowly, but in 16 of its 20 runs with four or more layers it is still rising at the end of training.

I WHEN HINDSIGHT GOALS ARE ALREADY SATISFIED

In the arm manipulation tasks of JaxGCRL, the goal is the position of an object, and this object rarely moves before the arm has learned to push it. As a result, most relabeled goals are already reached at the sampled state. Indeed, the left panel of Figure 14 shows that only 3 to 4% of the relabeled examples have a positive goal-reaching time, compared to 9 to 34% in the locomotion and navigation tasks. With so few positive goal-reaching times, the time loss and the hazard likelihood receive few informative targets. Consistent with this, the right panel of Figure 14 shows that the survival critics of ChronoSRL and SRL stay below a ToG of 100, whereas the contrastive critics of CRL and AC-CRL reach several hundred. Goals that also contain the position of the gripper, rather than that of the object alone, could make progress visible long before the object moves.

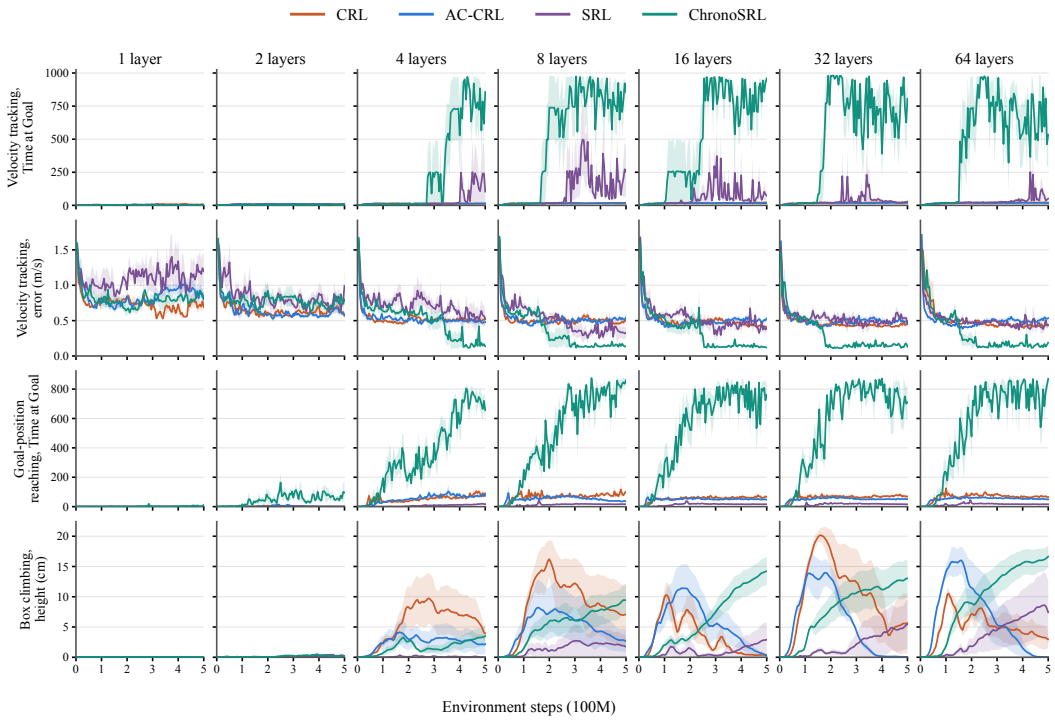


Figure 13: From four layers on, ChronoSRL learns to stay at velocity and position goals, and its box curriculum keeps rising until the end of training, while those of the contrastive baselines peak early and fall back. Rows show the four metrics of Figure 6 and columns the network depth, and each curve shows the mean over four seeds with one standard error over 500 million steps.

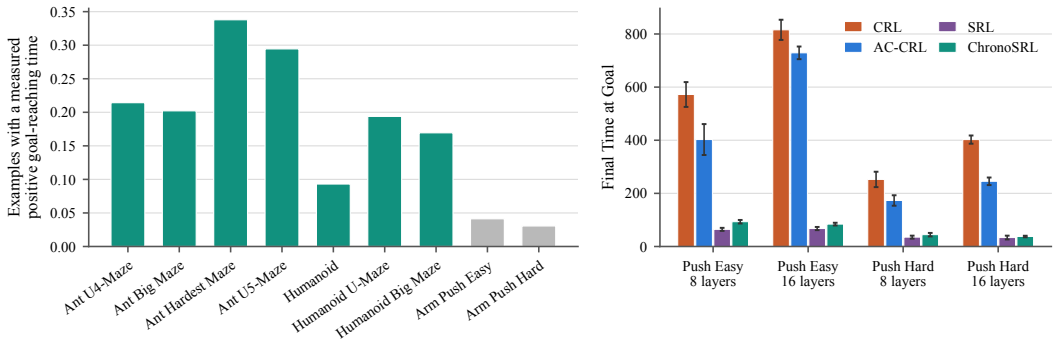


Figure 14: Temporal supervision needs trajectories that make progress toward their relabeled goals. The left panel shows the fraction of relabeled examples of ChronoSRL with a positive goal-reaching time in the final training batches, averaged over all runs of a task, and the right panel shows the final ToG on the two arm tasks. SRL uses its published arm settings, namely a dwell of $k = 20$ on Arm Push Easy and a discount of 0.99 on Arm Push Hard. Bars show the mean over four seeds with one standard error.