Keyframe Sampling, Optimization, and Behavior Integration: Towards Long-Distance Kicking in the RoboCup 3D Simulation League

Mike Depinet, Patrick MacAlpine, and Peter Stone

Department of Computer Science, The University of Texas at Austin
{msd775,patmac,pstone}@cs.utexas.edu

Abstract. Even with improvements in machine learning enabling robots to quickly optimize and perfect their skills, developing a seed skill from which to begin an optimization remains a necessary challenge for large action spaces. This paper proposes a method for creating and using such a seed by i) observing the effects of the actions of another robot, ii) further optimizing the skill starting from this seed, and iii) embedding the optimized skill in a full behavior. Called KSOBI, this method is fully implemented and tested in the complex RoboCup 3D simulation domain. To the best of our knowledge, the resulting skill kicks the ball farther in this simulator than has been previously documented.

1 Introduction

Every optimization needs a starting point. If the starting point is not in a region of the search space with a meaningful gradient, optimization is unlikely to be fruitful. For example, if trying to maximize the speed of a robot's walk, the robot has a much greater chance of success if given a stable walk to begin with. We refer to the starting point of an optimization for skill learning as a *seed skill*. Even with improvements in optimization processes, developing seed skills remains a challenge.

Currently most seed skills are written by hand and then tuned by a human until they resemble the desired skill enough to begin an optimization. Some seeds can also be acquired by having a robot mimic a human in a motion capture suit [1, 2]. We propose a third way of creating a seed, called *keyframe sampling*, which uses learning by observation. In this case, a robot observes the effects of actions of another object, and does its best to reproduce those effects. In our work robots observe another robot with the same model, although in principle this methodology could be applied to robots with different models or to humans using transfer learning ([3]) and different body mappings as described in [1, 2].

This paper considers a 3-step methodology of keyframe sampling, optimization, and behavior integration (*KSOBI*), which guides the development of a skill from watching a teacher to using the skill as part of an existing behavior. First, we describe KSOBI in Section 2, focusing on the keyframe sampling (KS) step.

We introduce the robot soccer domain in Section 3, and apply keyframe sampling and optimization (O) to kicking in robot soccer in Section 4. The robot soccer domain has the added complication that a skill is only useful if it can be incorporated into the robot's existing behavior. We describe behavior integration (BI) for our application, in Sections 4.5 and 4.6, and conclude with a summary and future work in Section 6.

2 KSOBI Overview and Keyframe Sampling

The goal of KSOBI's KS step is to use observations of another robot to quickly create an imitation skill. This imitation will later be used as the seed for an optimization, the O step, to create an optimized skill, which hopefully matches or improves upon the observed skill. Finally, that skill will be incorporated into the robot's existing behavior during the BI step. KSOBI is outlined in Figure 1.

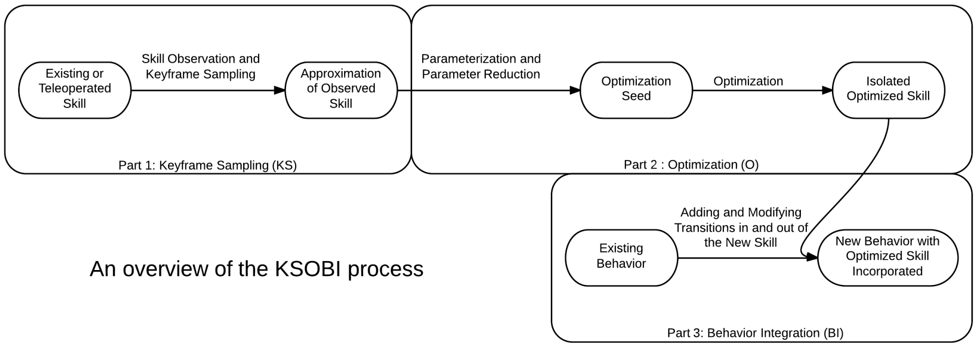
Existing or Teleoperated Skill
Skill Observation and Keyframe Sampling
Approximation of Observed Skill
Parameterization and Parameter Reduction
Optimization Seed
Optimization
Isolated Optimized Skill
Part 1: Keyframe Sampling (KS)
Part 2 : Optimization (O)
Existing Behavior
Adding and Modifying Transitions in and out of the New Skill
New Behavior with Optimized Skill Incorporated
An overview of the KSOBI process
Part 3: Behavior Integration (BI)

Fig. 1. An outline of KSOBI

Keyframe sampling assumes that the actions of the observed robot have observable effects. In the case of a physical robot, the robot's actions could be the torque applied to each motor while the effects are the change in rotation of joints. From the observed effects, a *keyframe skill* can be created directly.

A *keyframe* is defined to be a complete description of joint angles, either in absolute values or relative to the previous keyframe, with a *scale* for each joint indicating the percentage of the motor's maximum torque allowed to be used to reach the target angle. (The torque applied at any point in time is determined by a controller - often a PID controller, as in this work - but is multiplied by this value to affect how quickly a target angle is achieved.) A keyframe $k \in \mathcal{K} := \mathbb{R}^n \times \mathbb{R}^n \times \{0, 1\}$ where n is the number of joints, 0 indicates absolute angles, and 1 indicates relative angles.[1] The first n-vector gives target angles for each joint, while the second n-vector gives their scales. For example, the 3 joint keyframe $k_1 = ((0, 0, 0), (0.5, 0.5, 0.5), 0)$ indicates all joints should

[1] Note that in many robotic domains, including robot soccer, the distinction between relative and absolute joint positions is unnecessary since the joints have a specified non-overlapping range of possible values. This fact simplifies the above process since all keyframes may be unambiguously absolute.

be set to 0^o using half maximum torque, while $k_2 = ((180, 180, 0), (1, 1, 1), 1)$ indicates that the first and second motors should be rotated 180^o with maximum torque.

A *keyframe skill* (or *skill* unless otherwise noted) is defined as a list of keyframe-time pairs, where the time indicates how long to hold the paired keyframe. A skill $s \in (\mathcal{K} \times \mathbb{R})^m$ where m is the number of keyframes in the skill, and each of the m (k, t) pairs indicates that keyframe k should be the target for the next t seconds. For example, using k_1 and k_2 as defined above, the skill $s1 = ((k_1, 1.0), (k_2, 1.0))$ would indicate that the robot should take 1 second to get all its joints to 0^o (using at most half their torque) if possible, remaining there until 1 second has expired if time remains, then take another second to rotate joints 1 and 2 by 180^o as quickly as possible.

If the joint angles of an observed robot are directly observable, then a skill can be generated by recording each joint angle at specified time steps. This idea is the heart of keyframe sampling, which is made rigorous in the pseudocode below, where n is the number of joints in the robot model and T is the total time required by the skill divided by the time step:

```
define angle $\theta_{j,t}$ for $j \in [1, n] \cap \mathbb{Z}$ and $t \in [0, T) \cap \mathbb{Z}$
define keyframe $k_t$ for the same values of $t$
skill observeSkill(robot teacher, duration timeStep):
   int t = 0
   repeat:
     sampleKeyframe(teacher, t++)
     wait(timeStep)
   until teacher.skill.isDone()
   skill $s = ((k_0$,timeStep$), (k_1$,timeStep$), ...(k_{T-timeStep}$,timeStep$))$
   return skill
void sampleKeyframe(robot teacher, int t):
   for joint $j$ in teacher:
     $\theta_{j,t} = j$.angle
   if $t == 0$:
     $k_t = ((\theta_{1,t}, \theta_{2,t}, ..., \theta_{n,t}), (1, 1, ..., 1), 0)$
   else:
     $k_t = ((\theta_{1,t} - \theta_{1,t-1}, \theta_{2,t} - \theta_{2,t-1}, ..., \theta_{n,t} - \theta_{n,t-1}), (1, 1, ..., 1), 1)$
```

Using this method, the skill s will assume the observed starting position and, at each time step, attempt to assume the next set of observed joint angles as quickly as possible, imitating the observed object.

The generated skill s likely will not replicate the observed skill exactly, since it is just a sampling of several points in a presumably continuous motion, as described in more detail in [6]. However, as will be seen in Section 4, it may be close enough to use as the seed for an optimization. The hope is that the optimization will overcome the discontinuities in the seed skill s and create a skill which replicates or improves upon the observed motion.

Prior to optimization, it is necessary to parametrize the generated skill, allowing each value set by keyframe sampling to be varied by the optimization.

Often it will then be necessary to freeze a subset of the parameters, preventing them from changing during the optimization and reducing the dimension of the parameter space. Parameter reduction is addressed in Section 4.2. Having chosen which values may vary, a fitness function should be chosen and the optimization may begin. The optimization process is described in Section 4.3. Finally, once the new skill has been optimized in isolation, it must be incorporated into existing behavior. We illustrate the behavior integration (BI) process as applied to our application in Sections 4.5 and 4.6.

3 Application Domain: RoboCup 3D Simulation League

The target application for this work is the 3D Simulation RoboCup domain. In the 3D Simulation League simulated Nao robot agents play 11 vs 11 soccer in a physically realistic environment. The field is a 30m x 20m scale model of a full sized human field, and the robots are about 0.5 meters tall. Every 0.02 seconds, agents respond to information given from a game server with the torque to apply to each of their 22 motors.

In recent years, the RoboCup 3D Simulation League has been won primarily by creating fast and robust walks ([7], [8]). However, teams are now developing their own kicks ([9], [10], [11]). Kicking is difficult for three main reasons. First, robust kicking requires a smooth transition from walking and most walks involve some noise in reaching a target point. Second, kicking requires high precision in that a difference of a couple degrees on any joint in any keyframe will likely result in a failed kick. Third, there are many joints involved in a long distance kick and there are many keyframes between planting the foot and kicking the ball. This complexity results in a large search space for optimal kicks. Existing machine learning techniques help alleviate some of these problems, but there remains a need for finding reasonable starting seeds to guide the search through such a large parameter space, as well as a methodology for incorporating the resulting optimized skill into a full behavior, as is provided by KSOBI.

4 Learning to Kick from a Fixed Point

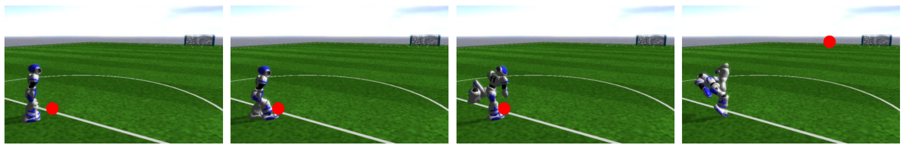

Fig. 2. The observed kick. Video available at: `http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/AustinVilla3DSimulationFiles/2014/videos/FCPKick.ogv`

We begin learning a kick under the assumption of a chosen starting location. In the RoboCup domain, an agent can expect this situation for its own kickoff. To learn a kick skill for a fixed starting location, we observe the previously furthest documented kick, belonging to FC Portugal [12] (see Figure 2). Using keyframe sampling, we create an approximation of this kick, which we use as a seed for optimization. The result of this optimization is the new longest known kick in the RoboCup 3D simulation environment.

4.1 Observing a Seed: Keyframe Sampling

The RoboCup server currently only provides the location of the head, torso, each leg, and each arm of robots to observers. This is not enough information to mimic another robot since there are multiple sets of joint angles that give the same locations for each body part. To solve this problem, we modify the server such that observers receive all of the joint angles of the observed robot, as required for keyframe sampling. The joint angles could reasonably be estimated by a real robot watching another real robot, so it seems like a reasonable level of detail to request. With this added information, we apply keyframe sampling at 16.67Hz (every 3 server cycles). Higher sampling rates give a seed skill more similar to the observed skill, but also result in more parameters to optimize. The 16.67Hz rate gave a sufficiently similar skill while keeping the parameter space reasonable in this case. As expected, the result is not an exact match of the observed skill. The imitation skill results in the robot kicking the ground behind the ball and falling over (Figure 3). However, the imitation is close enough to use as a seed for the optimization.

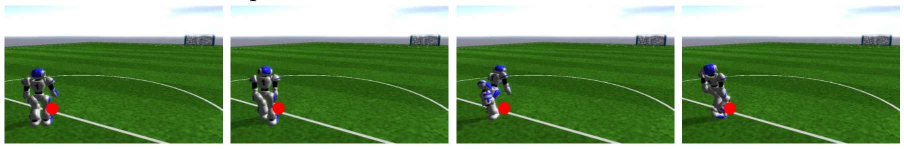

Fig. 3. The seed after observation. Video available at: `http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/AustinVilla3DSimulationFiles/2014/videos/InitialKick.ogv`

4.2 Fixed Point Training

The observed seed in Section 4.1 results in a skill with 89 key frames, each with every joint included, giving a total of 1958 parameters to train. Although this search space is prohibitively large, many of the parameters can be safely ignored. In fact, there is a need for parameter reduction before optimization in general when the seed is created by keyframe sampling. The goal in parameter reduction is to freeze parameters whose values will not affect the skill, removing them from the optimization and reducing the dimension of the search space. In addition to domain heuristics, seeds from keyframe sampling offer some general heuristics. First, any joint that does not change significantly between two keyframes can be fused between frames. Second, beginning and ending keyframes can sometimes be removed entirely. In this case, it is important to be careful not to disrupt the transition in and out of the skill (e.g. you would not want to remove keyframes responsible for setting the plant foot from a kicking skill).

In the case of our kicking seed, removing the head joints (a domain heuristic), any joint that does not change by more than 0.5 degrees between two frames, and the keyframes before the plant foot is set limits the skill to only 59 parameters. Adding 3 parameters for the starting location (x, y, and angle), results in 62 parameters to optimize. This is still a large state space, but it is manageable.

With the optimization parameters chosen, the next step is to define a fitness function. We use the distance traveled by the ball

$$fitness_{initial} = \begin{cases} -1 & : \text{ Failure} \\ finalBallLocation.x & : \text{ Otherwise} \end{cases}$$

where a "Failure" is any run in which the robot falls over, kicks backward, or runs into the ball before kicking it.

4.3 Optimizing with CMA-ES

Optimizing a set of parameters is the same as finding the global maximum of a fitness function $g(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ where n is the number of parameters being tuned. For optimization we use Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [13] as we have had previous success using CMA-ES for optimizing a walk, as documented in [14].

4.4 Fixed Point Results

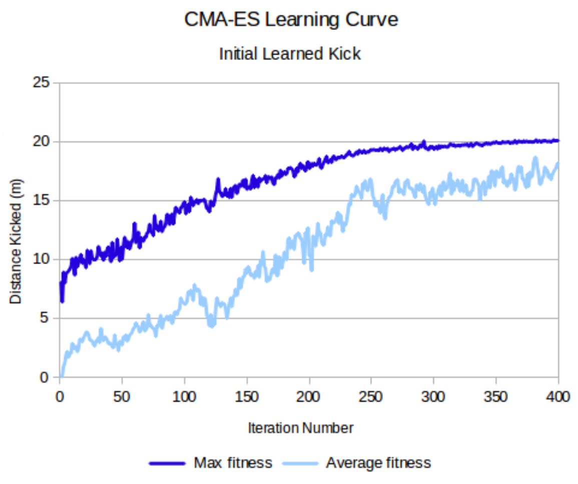

Fig. 4. CMA-ES Learning Curve for Initial Kick

After 400 iterations of CMA-ES with a population size of 200, the resulting skill is able to kick the ball 20 meters on average (see Figure 4). In addition to solving the problem of the robot kicking the ground behind the ball and helplessly falling over, the optimization produced a kick which exceeded the length of the original observed kick by more than 5 meters (Figure 5)!

We also consider two other fitness functions, producing slightly different resulting kicks. The first is a fitness function centered around accuracy. This function uses the same ball distance fitness as before except with a Gaussian penalty

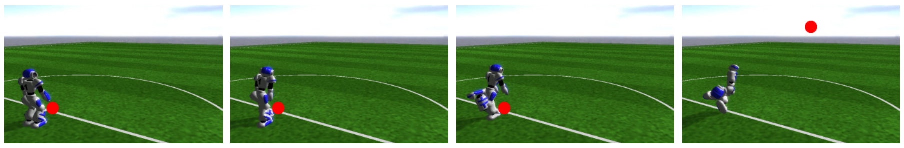

Fig. 5. The first learned kick. Video available at: `http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/AustinVilla3DSimulationFiles/2014/videos/LearnedKick.ogv`

for the difference between the desired and actual angles. Optimization with this function gives the powerful and predictable kick seen in Figure 6.

$$f_{accuracy} = \begin{cases} -1 & : \text{ Failure} \\ finalBallLoc.x * e^{-angleOffset^2/180} & : \text{ Otherwise} \end{cases}$$

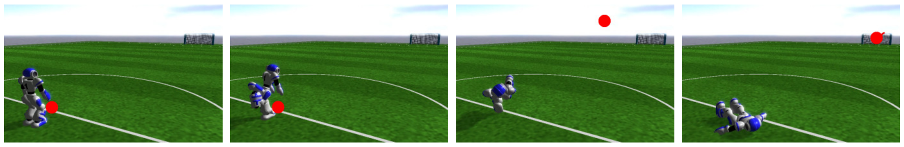

Fig. 6. Improved accuracy. Video available at: `http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/AustinVilla3DSimulationFiles/2014/videos/AccuracyKick.ogv`

The second is a fitness function centered around distance in the air. As the idea is to kick the ball long distances above opponents' heads, the fitness function heavily rewards the distance traveled by the ball before descending to 0.5m above the ground. It also rewards total distance and heavily penalizes missing the goal (ignoring any other tests of accuracy). Optimization with this function results in a noisy kick, but one that travels over 11m in the air (see Figure 7).

$$f_{air} = \begin{cases} -1 & : \text{ Failure} \\ 0 & : \text{ Missed goal} \\ 100 + finalBallLoc.x + 2 * airDist & : \text{ Otherwise} \end{cases}$$

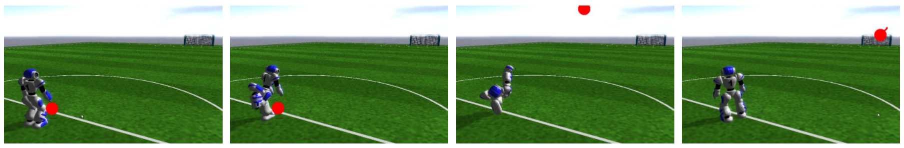

Fig. 7. Increased air distance. Video available at: `http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/AustinVilla3DSimulationFiles/2014/videos/AirDistKick.ogv`

These results are summarized in Table 1. All kicks are executed by the original NAO model used in the RoboCup 3D simulation for ease of comparison. In addition to being the longest documented kicks to our knowledge, these kicks are also the first ones able to score from any point in the offensive half of the field.

Table 1. Kick distances

Kick	Avg Distance (m)	Notes
Observed seed	About 15	
FCPortugal	About 17	Based on empirical data and verbal confirmation
Learned Kick	20.0(±0.12)	
Accuracy Kick	18.8(±0.29)	With placement $1.3^o(\pm 1.78^o)$ from target angle
Air Distance Kick	19.2(±0.38)	With $11.4m(\pm 0.25m)$ higher than 0.5m

4.5 Multi-Agent Training

With the kick optimized in isolation, we continue now to the behavior integration (BI) step. As the kick was optimized from a fixed point, integration into a legal kickoff is a natural first integration.

Unfortunately, scoring from the kickoff is illegal unless someone else touches the ball first in soccer. To rectify this, we introduce another agent with another skill which moves the ball as little as possible then gets out of the way. After optimizing this skill alone, using the server's play mode and the distance of the ball's movement to determine fitness, we optimize the touch and kick together, using the same fitness function as used for the kicker with an added penalty for either agent missing the ball or the kicker hitting the ball before the toucher.

$$f_{touch} = \begin{cases} -1 & : \text{ Failure} \\ 10 - finalBallLoc.magnitude & : \text{ Otherwise} \end{cases}$$

where for this single case, a "Failure" is when the robot falls over, fails to touch the ball, or touches the ball more than once.

$$f_{kickoff} = \begin{cases} -1 & : \text{ Failure} \\ -1 & : \text{ Wrong touch order} \\ -1 & : \text{ Either agent missed} \\ 100 + finalBallLoc.x + 2{*}airDist & : \text{ Otherwise} \end{cases}$$

4.6 Multi-Agent Results

After a successful 400 iteration optimization with population size of 150 (see figure 8), two agents are able to legally and reliably score within 8 seconds of the game starting and within 3 seconds of the ball first being touched (see figure 9). Adding this kickoff behavior and changing nothing else dramatically improves the team's overall performance (see Table 2). If in addition we alter the agent to improve the accuracy resulting from the server's beam command before kickoffs (and re-run the multi-agent optimization with this improvement), we get the scoring percentages presented in Table 3.

The percentage of kickoffs which score varies with the opponent team. Most kickoffs which fail to score are a result of opponent player formation. We have not found a kick that makes it all the way to the goal in the air, so a player

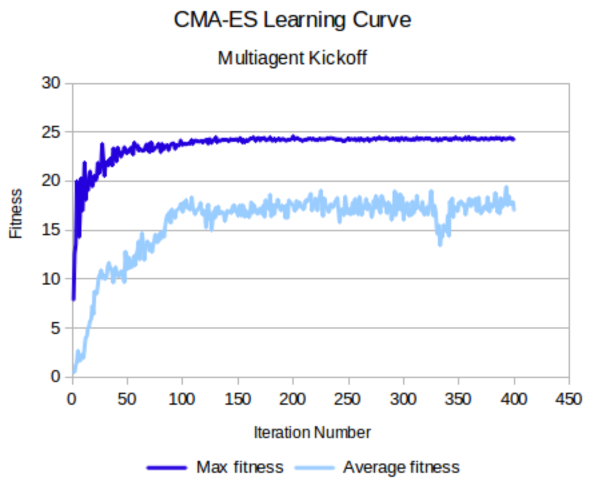

Fig. 8. CMA-ES Learning Curve for Multiagent Kickoff

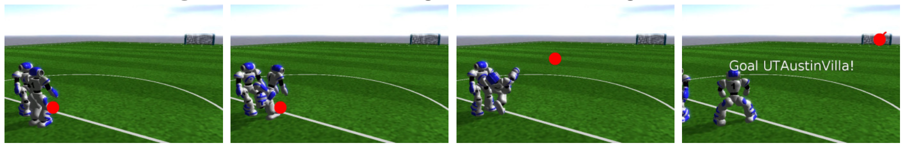

Fig. 9. Multiagent Kickoff. Video available at: `http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/AustinVilla3DSimulationFiles/2014/videos/Kickoff.ogv`

located where the ball bounces on its path to the goal effectively stops kickoff goals. That said, the location at which the ball bounces is not always the same. In the future, the kicking agent could have multiple kickoff kicks and trajectory information for each and could use that information at run-time to choose a kick that misses opponents.

5 Related Works

Other forms of learning from observation have been explored, and are described in [4]. In the context of that review, keyframe sampling has several qualities. First, we may use either a human or a robot for teaching, whereas most approaches require a human teacher. Secondly, the data set from which we learn is limited to a single sequential series of observed actions, rather than a larger set covering more initial state spaces and possibly providing repetition. It should also

Table 2. Game statistics

Using Kickoff	Opponent	Average Goal Differential	W-L-T
No	FCPortugal	0.335 (+/-0.023)	368-85-547
Yes	FCPortugal	1.017 (+/-0.029)	775-6-219
Yes	WithoutKickoff	0.385 (+/-0.022)	416-44-540

Table 3. Percentage of scored kickoffs against the top 4 finishers from RoboCup 2013

Opponent	Beginning of Half	During Half
FCPortugal	92.19%	77.15%
UTAustinVilla	76.70%	54.15%
SeuJolly	77.00%	77.66%
Apollo3D	89.30%	65.60%

be noted that our methodology is assuming a continuous state space (discrete time, continuous space) and our policy derivation is completed by a mapping function (the identity map in this case since the observed robot has the same model as the learning robot). Finally, it is important to note that the policy we derive is intended only as the seed for further optimization. While initially the policy developed from observation may not perform as well as the observed policy from which it is derived, in general we expect the learned policy's performance after optimization to surpass that of the original observed policy.

Another approach which uses learning by observation to begin an optimization is described in [2]. In that work, a human's motions are captured via a Microsoft Kinect, converted to a robot model, and optimized to achieve balance despite differences in joints and mass distributions. This is similar to the KS and O steps of KSOBI, with some important differences in the sampling rate and the parameter reduction steps. In this work, the authors sample at 50Hz then model the motion of each joint as a function in order to hopefully reduce the search space of the optimization. In our application of KSOBI, we use a smaller sampling rate (5-17Hz) and use several heuristics to further reduce the parameter search space. Additionally, this work requires a human teacher, whereas KSOBI can use either a robot or a human teacher.

In [5], the authors describe two methods of having humans teach robots by demonstration. One way is to record the robots motion as a human moves it (trajectory-based). The other (keyframe-based) involves inferring the trajectory between two points set by a human. The authors try to make the interaction as easy as possible for humans by allowing a hybrid approach, in which either trajectory-based or keyframe-based methods may be recorded. This methodology is similar to our keyframe-sampling approach, except that our approach does not require a human teacher, and instead records keyframes at a certain rate instead of at important points. Depending on the sampling rate, keyframe-sampling could be more similar to the trajectory-based approach described in this article. Moreover, this article expects to create a new behavior by learning from demonstration alone, while in KSOBI, demonstration is only the first step.

6 Summary and Future Work

This paper introduced the KSOBI process, guiding the development of a skill from watching another robot (keyframe sampling - KS), to optimizing the resulting sampled skill (optimization - O), to integrating the optimized skill into an existing behavior (behavior integration - BI). The full KSOBI process was applied in a complex simulated domain. Along the way, we showed the success of this method with a set of new kicks which raise the bar for how far agents can kick in the RoboCup 3D simulation league.

Future work includes another desirable behavior integration, which requires being able to approach the ball before kicking it. Such an integration would allow the kick to be used for shooting and passing during regular gameplay. Both the approach and the kick would need to be quick for the kick to be useful during a game. Although we've made progress on this front ([6]), it remains an important area for future work.

It may be possible to make more robust kicks by defining them using trajectories relative to the ball instead of fixed joint angles. The UTAustinVilla codebase already has a set of kicks parametrized by trajectories relative to the ball [11], however they seldom exceed a distance of 5 meters. Future work may also include finding a happy medium between the flexibility of those kicks and the distance achieved by fixed joint angle kicks.

References

1. Setapen, A., Quinlan, M., Stone, P.: Marionet: Motion acquisition for robots through iterative online evaluative training. In: Ninth International Conference on Autonomous Agents and Multiagent Systems - Agents Learning Interactively from Human Teachers Workshop (AAMAS - ALIHT). (2010)
2. Seekircher, A., Stoecker, J., Abeyruwan, S., Visser, U.: Motion capture and contemporary optimization algorithms for robust and stable motions on simulated biped robots. In Chen, X., Stone, P., Sucar, L., van der Zant, T., eds.: RoboCup 2012: Robot Soccer World Cup XVI. Volume 7500 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2013) 213–224
3. Taylor, M.E., Stone, P.: Transfer learning for reinforcement learning domains: A survey. Journal of Machine Learning Research **10** (2009) 1633–1685
4. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robotics and autonomous systems **57** (2009) 469–483
5. Akgun, B., Cakmak, M., Jiang, K., Thomaz, A.: Keyframe-based learning from demonstration. International Journal of Social Robotics **4** (2012) 343–355
6. Depinet, M.: Keyframe sampling, optimization, and behavior integration: A new longest kick in the robocup 3d simulation league. Master's thesis, University of Texas at Austin (2014) Undergraduate Thesis.
7. Bai, A., Chen, X., MacAlpine, P., Urieli, D., Barrett, S., Stone, P.: Wright Eagle and UT Austin Villa: RoboCup 2011 simulation league champions. In Roefer, T., Mayer, N.M., Savage, J., Saranli, U., eds.: RoboCup-2011: Robot Soccer World Cup XV. Lecture Notes in Artificial Intelligence. Springer Verlag, Berlin (2012)
8. MacAlpine, P., Collins, N., Lopez-Mobilia, A., Stone, P.: UT Austin Villa: RoboCup 2012 3D simulation league champion. In Chen, X., Stone, P., Sucar, L.E., der Zant, T.V., eds.: RoboCup-2012: Robot Soccer World Cup XVI. Lecture Notes in Artificial Intelligence. Springer Verlag, Berlin (2013)
9. Ferreira, R., Reis, L., Moreira, A., Lau, N.: Development of an omnidirectional kick for a nao humanoid robot. In Pavón, J., Duque-Méndez, N., Fuentes-Fernández,

R., eds.: Advances in Artificial Intelligence ? IBERAMIA 2012. Volume 7637 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 571–580

10. Cruz, L., Reis, L., Lau, N., Sousa, A.: Optimization approach for the development of humanoid robots? behaviors. In Pavn, J., Duque-Mndez, N., Fuentes-Fernndez, R., eds.: Advances in Artificial Intelligence ? IBERAMIA 2012. Volume 7637 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 491–500
11. MacAlpine, P., Urieli, D., Barrett, S., Kalyanakrishnan, S., Barrera, F., Lopez-Mobilia, A., Ştiurcă, N., Vu, V., Stone, P.: UT Austin Villa 2011: A champion agent in the RoboCup 3D soccer simulation competition. In: Proc. of 11th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS). (2012)
12. Lau, N., Reis, L.P., Shafii, N., Ferreira, R.: Fc portugal 3d simulation team: Team description paper. In: RoboCup 2013. (2013)
13. Hansen, N.: The CMA Evolution Strategy: A Tutorial. (2009) `http://www.lri.fr/~hansen/cmatutorial.pdf`.
14. MacAlpine, P., Barrett, S., Urieli, D., Vu, V., Stone, P.: Design and optimization of an omnidirectional humanoid walk: A winning approach at the RoboCup 2011 3D simulation competition. In: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI). (2012)