EPiC Series in Computing

Volume 50, 2017, Pages 239–245

GCAI 2017. 3rd Global Conference on Artificial Intelligence

Optimizing Kick Trajectory: A Comparative Study

Pedro Peña and Joseph Masterjohn and Ubbo Visser

University of Miami, Department of Computer Science,
1365 Memorial Drive, Coral Gables, FL, 33146, USA
E-Mail: {pedro|joe|visser}@cs.miami.edu

Abstract

Incorporating a dynamic kick engine that is both fast and effective is pivotal to be competitive in one of the world's biggest AI and robotics initiatives: RoboCup. Using the NAO robot as a testbed, we developed a dynamic kick engine that can generate a kick trajectory with an arbitrary direction without prior input or knowledge of the parameters of the kick. The trajectories are generated using cubic splines (two degree three polynomials with a via-point), cubic Hermite splines or sextics (one six degree polynomial). The trajectories are executed while the robot is dynamically balancing on one foot. Although a variety of kick engines have been implemented by others, there are only a few papers that demonstrate how kick engine parameters have been optimized to give an effective kick or even a kick that minimizes energy consumption and time. Parameters such as kick configuration, limits of the robot, or shape of the polynomial can be optimized. We propose an optimization framework based on the Webots simulator to optimize these parameters. Experiments on different joint interpolators for kick motions have been observed to compare results.

1 Motivation and Background

Generating dynamic motions on a robot without explicitly programming the motion is a difficult task and a fairly new research area. Kick engines such as [2, 12, 8, 7, 10, 3] have been developed for the NAO robot and although they are dynamic, there are static values incorporated into the kicks such as retraction points, foot positions from floor, hip/ankle pitch and hip/ankle roll ratio, and shapes of trajectories. These values are usually derived from empirical observations and are not guaranteed to be optimal values.

The difficulty of the task is to optimize parameters on the physical robot because the robot is limited by hardware, energy consumption, and most importantly real time execution. Hence, the robot cannot run for thousands or even millions of iterations to get a good set of parameters. Other dynamic kick engines such as [13] developed a kick engine for the THOR-MANG from Robotis that generate static kick motions. Lengagneua et al. [6] have developed a kick engine that incorporates optimization offline and a re-planning step when the kick is executed but do not optimize on the physical robot or use a simulation software that takes into account hardware properties of the robot.

Wenk et al. [10] developed a kick engine that generates online kick motions using trajectories generated by Bézier curves which are continuously differentiable.

C. Benzmüller, C. Lisetti and M. Theobald (eds.), GCAI 2017 (EPiC Series in Computing, vol. 50), pp. 239–245

Böckmann et al. [3] provided a mass spring damper model to model motor behavior. The authors also adapted Dynamic Motion Primitives (DMP) to generate kick trajectories. Kick trajectories are usually generated from Bézier curves or via-point kick trajectories, but here the authors used a PD controller with a forcing term in the transformation system to control the shape of the trajectory.

Sung et al. [9] used full body motion planning and via-point representation to generate joint angle trajectories. These trajectories are generated using five degree polynomials and via points are specified to constrain the swing trajectory. In order to create efficient full body motion trajectories, the author used optimization techniques such as Semi-Infinite Programming (SIP) to specify constraints such as minimal energy and torque. The optimization also dealt with joint redundancy.

Muhammad Usman et al. [1] used RoboCup Soccer Simulation 3D to optimize Bézier curves and cubic Hermite splines. The optimization method used by the authors was Particle Swarm Optimization. The simulator used for these experiments is mostly for agents and it is not transferable to physical robots because the kinematics of the robot are different as well as the dynamics produced by the simulator. In contrast, the Webots simulator used in this framework is directly related to the physical robot, and the optimized parameters from the simulator can be directly used with minimal modifications to compensate for each physical robot.

Jouandeau et al. [5] generated rocking motions for the swinging leg, and also created a swing motion for the torso to generate kicks. The authors used an optimization method called Confident Local Optimization techniques (CLOP) for the swinging foot and torso. The kicks were tested in the RoboCup 3D simulator.

Yi et al. [13] used THOR-OP (Tactical Hazardous Operations Robot - Open Platform) full sized humanoid robot to generate kick and walk motions for the AdultSize League in RoboCup 2014. The kick motions generated were handled by the hybrid walking controller to create smooth transitions between the dynamic walk and strong kick.

We propose a new kick engine for the NAO robot that can generate kick trajectories using cubic splines, cubic Hermite splines, and sextic polynomials (six degree polynomials). A comparision of these kick trajectories generated by different interpolators will be compared to observe the best kick trajectories generated by each interpolator. The values of the kick trajectories are optimized in the Webots simulator to get a good set of parameter values. The overview of the control framework is presented in fig. 1.

In this paper, the optimization module of the framework will be explained which includes

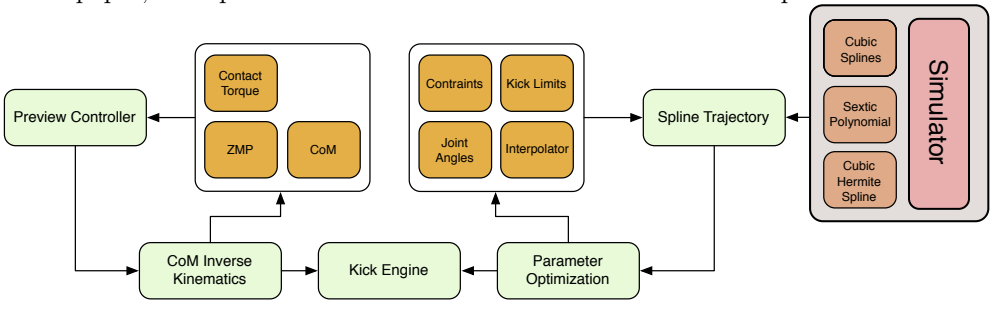

Figure 1: Overview of the control framework for the kick engine

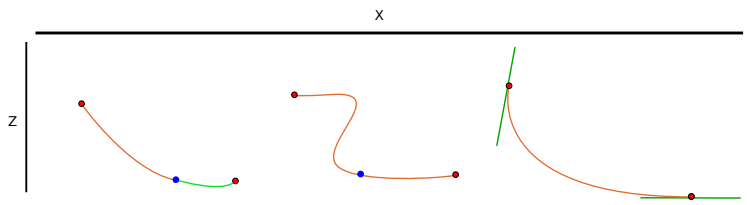

Figure 2: The left polynomials are the cubic splines. As shown in the figure, two polynomials need to be defined with a knot referred to as the via-point. The middle polynomial is referred to as a sextic (six degree polynomial). The six degree polynomial allows definition of a via point without adding another polynomial. The right polynomial is a cubic Hermite spline which guarantees the control points to be tangent to the tangent vectors creating smooth trajectories for the kick. In this case, no via-point is defined unless a cubic Hermite spline is chained with another.

the simulator and the kick interpolators. The paper is organized as follows: we describe how each interpolator is constructed from the properties of the kick trajectory in Section 2. The model optimization on the kick is discussed in Section 3, and our experiments and results are explained in Section 4, followed by the conclusion in Section 5.

2 Kick Trajectory

In order to generate a kick trajectory, the most trivial case is to change the leg joint angles until a desired kick configuration has been reached. This is tedious work and will be suboptimal since the joint space is very large. Another approach is to use optimization to find key-frame values, but this only works for a set of kicks and it is not dynamic enough to create any kick trajectory. Therefore, a more efficient solution is to generate motions using polynomials [11]. Polynomials are a great solution in robot motion because they can be configured to generate smooth curves. To generate a polynomial for the kick motion requires constraints such that the motion generated by the polynomial does not conflict with any unwanted configuration. The polynomials will not be used in the joint space, but rather will be used to determine the next position of the swinging foot. When the position of the foot is determined by the cubic polynomial, the Inverse Kinematic module will provide the angles for the foot position requested. In the case of cubic splines, two cubic polynomials are generated. The point where the two polynomials meet is called the via-point. The purpose of this point will be discussed later. The cubic polynomial is as follows,

$$\alpha_1(t) = a_{13}t^3 + a_{12}t^2 + a_{11}t + a_{10} \quad \alpha_2(t) = a_{23}t^3 + a_{22}t^2 + a_{21}t + a_{20}$$

In pursuance of generating an arbitrary motion, specific constraints need to be put upon the polynomials. Since there are two cubic polynomials (i.e. 8 coefficients / degrees of freedom), there are 8 constraints. The first constraint is the point of the first polynomial at $t = 0$. At $t = 0$, the kick motion will swing the leg back. This is called the retraction point. This is the point farthest from the ball. The second constraint is that the velocity of the first point at $t = 0$ which is zero. The third constraint is the position of the via-point where both cubic polynomials meet. The via-point is used to determine the height of the kick trajectory.

It is also a very important point because it is where both polynomials meet. Hence, both polynomials need to have the same position at this point and their velocities need to match. Moreover, the acceleration at the via point for both polynomials also needs to match. This guarantees a smooth trajectory with C^2 continuity. The last two constraints define the position and velocity of the second cubic spline at $t = t_f$. The constraints are summarized as follows,

$$
\begin{array}{rcll}
\alpha_1(0) & = & [x_0, y_0, z_0] & \text{(retraction point)} \\
\dot{\alpha}_1(0) & = & [0,0,0] & \\
\alpha_1(t_{via}) & = & [x_{t_{via}}, y_{t_{via}}, h_{floor}] & \text{(constraint on foot from floor)} \\
\alpha_2(0) & = & [x_{t_{via}}, y_{t_{via}}, h_{floor}] & \text{(constraint on foot from floor)} \\
\dot{\alpha}_1(t_{via}) & = & \dot{\alpha}_2(0) & \\
\ddot{\alpha}_1(t_{via}) & = & \ddot{\alpha}_2(0) & \\
\alpha_2(t_f) & = & [x_f, y_f, z_f] & \text{(contact point)} \\
\dot{\alpha}_2(t_f) & = & [0,0,0] &
\end{array}
\tag{9}
$$

Moreover, as shown above, the constraints are defined as vectors because there needs to be polynomials for the x, y, and z plane. The six cubic polynomials (two for each plane) will form a parametric curve in $\mathbb{R}^3$. To solve the coefficients of the polynomials we solve the following system which was created by inputting the constraint in the polynomial and rearranging terms. As seen in (9), a_{10}, a_{11}, and a_{20} do not need to be a part of the system of equations because their answer is trivial ($\alpha_1(0) = a_{10}, \dot{\alpha_1}(0) = a_{11}, \alpha_2(0) = a_{20}$).

Solving (9), gives us the coefficients for our polynomials, but we still are missing one step. Before we begin solving (9), we need to determine the optimal via point position. This is discussed in section 3. We can also do the same for sextic interpolation (six degree polynomial) as we did for the cubic polynomial with the advantage that there is only one equation for the interpolation rather than two such as the cubic spline. For the cubic Hermite spline, the end points referred to as the control points on the spline needs to be defined. In this case, it is the retraction point and the contact point as (9). Also, a tangent vector for each control point needs to be defined. The initial tangent vectors are suggested to be the vector from the retraction point to the contact point. In section 3, the optimizer will find the optimal tangent vectors. In fig. 2, a kick trajectory for each is shown.

3 Model Optimization

Generating a good parameter set is important to attain a good kick. Although these values can be found empirically, it is a tedious task. We therefore used the Covariance Matrix Adaptation Evolution Strategy (*CMA-ES*, [4]) for model optimization. The values were optimized and visualized in Webots, which can be seen as the standard simulation software for NAOqi, the robot's OS. For each kick trajectory, the whole parameter space was optimized. Therefore for the sextic and cubic trajectories, the retraction point, contact point, via point, and length of the kick were optimized. The time lapse to arrive at the via point was also optimized. The parameter space for sextic and cubic is 12-dimensional. For cubic Hermite spline, the retraction and contact points and their corresponding tangent vectors were also optimized. The dimension space of cubic Hermite spline is 14-dimensional. The extra two dimensions correspond to the length of the kick and a single parameter non-linear transformation of time to control speed along the path without affecting the shape of the curve. The initial population began with a feasible kick trajectory which was constructed from empirical observations. This allowed the optimization to finish much faster than if the seed was a random kick trajectory. The initial objective function of the optimization was the distance of the ball traveled when the ball was kicked. Using this objective function, the agent learned to use the momentum of it's body to kick the ball further while falling. Therefore, the objective function was updated with a penalty if the robot fell. In turn, the agent learned to kick farther without falling.

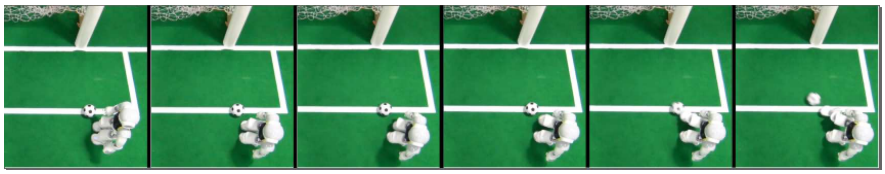

Figure 3: Diagonal kick generated by cubic spline interpolator on a NAO physical robot

4 Experiments

For the validation of our approach, various kicks (forward, side, diagonal, backward) were generated and data for kick trajectory generation was recorded. The first graph in fig. 4 shows the fitness function value and it demonstrates that the optimization was able to learn new kicks that can kick the ball further than the original kicks built from observation. It is also important to notice that sextic had a harder time to minimize the function value as well as Hermite. The cubic spline interpolator was able to converge faster as well as more stable at every generation. This can be due to the initial seed of the other interpolators being inferior to the seed of cubic spline making it harder to explore. On the right of the fitness function graph in fig. 4, the distance traveled of the ball was plotted. Initially, it can be seen that the ball traveled about three meters with a viable seed. After 300 iterations, the distance of the ball increased by a factor of about two. It is important to note that the distance traveled of the ball in each iteration is the maximum distance traveled for the current generation. Hence within each generation, the distance traveled for the population must have been less than or equal to the maximum distance traveled in the current generation. The dynamics of the parameter values for the three kick interpolators can also be seen in fig. 4. As can be seen, the optimization explores the parameter space vastly for the first 200 iterations. After exploring the parameter space for 200 iterations, the optimization stops searching and rather exploits the current parameters. In general, the optimization converges quickly because the initial seed of the optimization is a viable kick. Therefore this shows how essential it is to start with a good seed. The kick trajectories were also generated on the physical robots. A video of the kick engine can be found at: https://www.youtube.com/watch?v=4fmuqI_CpQw and a diagonal kick trajectory generated using cubic splines can be seen in fig. 3.

5 Conclusion

We have compared results of kick trajectories generated by different joint interpolators. The results have shown that kicks can be optimized using these interpolators. The joint interpolators used for these experiments were cubic splines which are two three degree polynomials that meet at a via point, a sextic polynomial which is a six degree polynomial, and a cubic Hermite spline which needs to have the end points and their tangent vectors defined. These results demonstrated that the joint interpolators can be optimized to get powerful kicks rather than optimizing on the full joint space; drastically reducing the dimension of the problem. The cubic spline resulted the best joint interpolator in terms of minimizing the function value at a faster pace. The results have also exhibited that with a good initial seed, the optimization can converge at a rather fast pace; in contrast with optimizations done on the joint space which can take longer training times. For future work, optimizations can be done in parallel with diverse feasible seeds to converge on suboptimal kicks and compare results.

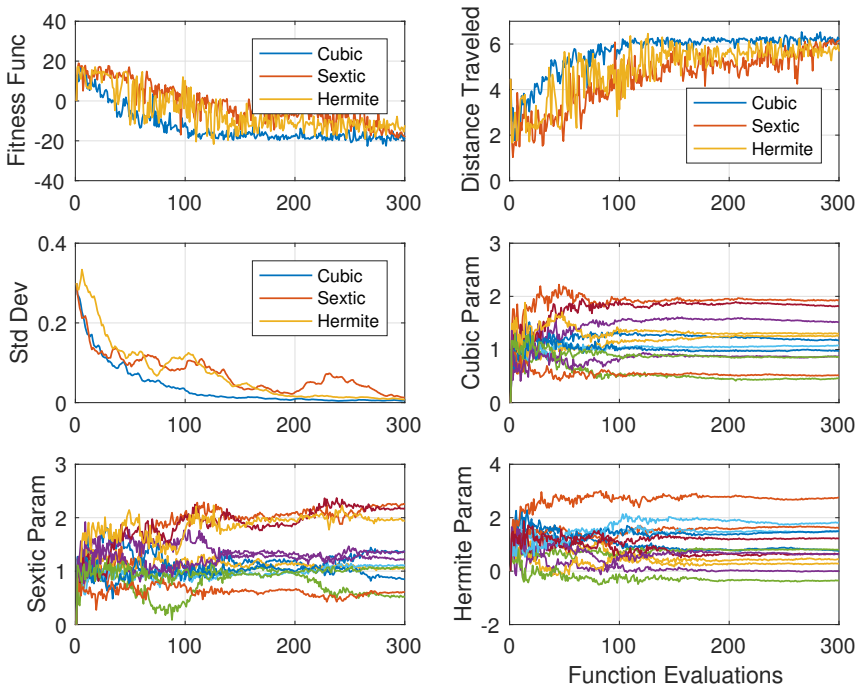

Figure 4: Results from optimization on Webots simulator

References

[1] Muhammad Usman Arif, Syed Ali Raza, and Sajjad Haider. On developing a hybrid approach for kick optimization in humanoid robots. In *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on*, pages 1512–1516. IEEE, 2014.

[2] Inge Becht, Maarten de Jonge, and Richard Pronk. A dynamic kick for the NAO robot. *Project Report, July*, 2012.

[3] Arne Böckmann and Tim Laue. Kick motions for the NAO robot using dynamic movement primitives. *arXiv preprint arXiv:1606.00600*, 2016.

[4] Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

[5] Nicolas Jouandeau and Vincent Hugel. Optimization of parametrised kicking motion for humanoid soccer player. In *Autonomous Robot Systems and Competitions (ICARSC), 2014 IEEE International Conference on*, pages 241–246. IEEE, 2014.

[6] Sébastien Lengagneua, Philippe Fraisse, and Nacim Ramdani. Planning and fast re-planning of safe motions for humanoid robots: Application to a kicking motion. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 441–446. IEEE, 2009.

[7] Judith Müller, Tim Laue, and Thomas Röfer. Kicking a ball–modeling complex dynamic motions for humanoid robots. In *RoboCup 2010: Robot Soccer World Cup XIV*, pages 109–120. Springer, 2011.

[8] Johannes Strom, George Slavov, and Eric Chown. Omnidirectional walking using ZMP and preview control for the NAO humanoid robot. In *Robot Soccer World Cup*, pages 378–389. Springer, 2009.

[9] Chang Hyun Sung, Takahiro Kagawa, and Yoji Uno. Planning of kicking motion with via-point

representation for humanoid robots. In *Ubiquitous Robots and Ambient Intelligence (URAI), 2011 8th International Conference on*, pages 337–342. IEEE, 2011.

[10] Felix Wenk and Thomas Röfer. Online generated kick motions for the NAO balanced using inverse dynamics. In *Robot Soccer World Cup*, pages 25–36. Springer, 2013.

[11] Robert L Williams. Simplified robotics joint-space trajectory generation with a via point using a single polynomial. *Journal of Robotics*, 2013, 2013.

[12] Yuan Xu and Heinrich Mellmann. Adaptive motion control: Dynamic kick for a humanoid robot. In *Annual Conference on Artificial Intelligence*, pages 392–399. Springer, 2010.

[13] Seung-Joon Yi, Steve McGill, Qin He, Dennis Hong, and D Lee. Walk and kick motion generation for a general purpose full sized humanoid robot. In *Workshop on Humanoid Soccer Robots, IEEE-RAS International Conference on Humanoid Robots (Humanoids). IEEE*, 2014.