Embedded Control Synthesis Using One-Step Methods in Discrete Mechanics

Jarvis Schultz and Todd D. Murphey

Abstract— **Low bandwidth control and estimation for nonlinear systems presents a challenging problem that is often encountered when dealing with implementation on an embedded platform. Discrete mechanics techniques for system modeling are well-suited to low-bandwidth applications because they posses desirable numerical properties over a large range of timesteps including exact constraint conservation, and excellent Hamiltonian and momentum behaviors. We present an overview of a variational integrator based discrete mechanics system representation and corresponding state choice that allows the discrete flow to be expressed as a one-step map as required by classical digital control design tools. This modeling paradigm is used to experimentally control an underactuated, nonlinear system with relatively low control frequency. Simulations of the experimental system demonstrate significantly better extended Kalman filter performance using the present framework over a traditional one-step Euler approximation.**

I. INTRODUCTION

With technological progress the capabilities of sensors, actuators and processors is continually progressing. Yet many situations benefit from utilizing less advanced components. Less capable components usually imply cheaper costs, and in industrial settings this becomes an important design consideration. Mathematical methods that allow one to accomplish control tasks with lower bandwidths, fewer components, and less precise sensors yields cheaper consumer goods.

In some instances, technological progress in one direction can mean a negative trade-off in another. The Microsoft Kinect® is a perfect example; several years ago, the idea of a very inexpensive consumer electronic providing real-time 3D point cloud information was out of the question. Now the community has embraced the vast amount of data that the Kinect provides. However, what if one desires to use this device to provide feedback of a highly dynamic system? The relatively low frequency (≈30 Hz) of the Kinect must be taken into consideration to obtain high-performance estimators and controllers.

Due to sensor noise, and unmeasurable states, estimation and filtering become a near-requirement in real-world control systems. Whether utilizing traditional Gaussian-assumption filters such as the Kalman Filter or numerically estimating uncertainty propagation through sampling-based methods such as a particle filter, an underlying mechanical model and numerical integrator that accurately reflects the embedded system's characteristics is crucial to the success of the filtering algorithm. As a digital embedded system will certainly be running the control/ estimation loop at discrete time increments, it is necessary to choose a discrete approximation of the inherently continuous system. Traditional choices include implicit and explicit Euler methods and Runge-Kutta methods; moreover, when dealing with an embedded system, it is desirable that the discrete approximation be an explicit, one-step method. A one-step method means that the system will need to remember less history of the of the state which means less memory usage and less computation. Additionally, standard discrete linear systems tools such as Linear Quadratic Gaussian (LQG) estimators, Linear Quadratic Regulator (LQR) controllers, and Kalman filters are often an important component of a control loop, even for nonlinear systems [7]. In the nonlinear setting, one typically uses a local linearization to achieve the standard form of a discrete linear system given by

$$x_{x+1} = A_k x_k + B_k u_k. \tag{1}$$

To use these traditional tools of discrete systems (LQR and LQG) with integration techniques that are greater than one-step methods one must convert the discrete representation into an equivalent one-step method which usually means introducing extra dimensions to the state representation of the system. As a result it is quite common, in practice, to restrict a system's discrete approximation to a simple, low-order method such as explicit Euler.

In highly dynamic systems or systems with holonomic constraints, simple Euler integration tends to artificially add or subtract energy to simulations; additionally for constrained systems, this integrator exhibits constraint drift which may eventually diverge to such an extent that the simulation goes unstable. As a consequence, this convenient and practical design choice may lead to poor estimator and controller performance, especially in low-bandwidth scenarios.

In this work, we present a variational integrator based, discrete mechanics modeling framework that addresses many of these aforementioned problems with the practical implementation of control schemes on real embedded systems. A brief overview of discrete mechanics is first presented in Section II, followed by a description of a particular choice of state that enables representing the system as a one-step

J. Schultz `jschultz@u.northwestern.edu`
T.D. Murphey `t-murphey@northwestern`
Department of Mechanical Engineering, Northwestern University, Evanston, IL

This material is based upon work supported by the National Science Foundation under award IIS-0917837. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

method in Section II-B. This state choice and modeling paradigm allows one to find exact an exact linearization of the discrete approximation to the system. Next, an example system and corresponding experimental apparatus that exhibits many of the aforementioned embedded system constraints is discussed in Section III. Finally, experimental and simulation-based results are presented illustrating the features of applying this modeling framework to an embedded system in Section IV.

II. DISCRETE MECHANICS

In the discrete mechanics framework, one attempts to find a sequence $\{(t_0,q_0),(t_1,q_1),\ldots,(t_n,q_n)\}$ that approximates a continuous time trajectory of a system i.e. $q_k \approx q(t_k)$ where t is time, and $q \in Q$, the configuration space of the system. Define the discrete Lagrangian as an approximation to the action integral over a single timestep of size $\Delta t = t_{k+1} - t_k$ as

$$L_d(q_k, q_{k+1}) \approx \int_{t_k}^{t_{k+1}} L(q(\tau), \dot{q}(\tau))d\tau. \tag{2}$$

Note that various choices for this approximation are available, and the particular choice will govern the accuracy order of the integrator. The next step is replacing the traditional action integral with an action sum

$$S(q[t_0,t_f]) = \int_{t_0}^{t_f} L(q(\tau),\dot{q}(\tau))d\tau \approx \sum_{k=0}^{n-1} L_d(q_k,q_{k+1}). \tag{3}$$

Applying a variational principle to find stationary values of this action sum yields an implicit set of difference equations referred to as the unforced, unconstrained Discrete Euler-Lagrange (DEL) equations[1]

$$D_1 L_d(q_k,q_{k+1}) + D_2 L_d(q_{k-1},q_k) = 0. \tag{4}$$

In these implicit difference equations, given two sequential configurations q_{k-1} and q_k, the next configuration in the sequence q_{k+1} is found by using a numerical root solver for Eq. (4).

If a system has forcing, time integrals of the continuous forces are approximated with numerical integration rules, and through a discrete analog to the Lagrange-D'Alembert principle, the forced DEL equations may be derived. Also note that if the system involves holonomic constraints, these are also easily incorporated into the DEL equations. Both of these additions simply add terms to the right-hand side of Eq. (4), but the terms on the left-hand side remain unchanged.

Given a constraint of the form $\phi(q(t)) = 0$, the discrete integrator exactly enforces $\phi(q_k) = 0$ at every step. In traditional integration schemes, such as Runge-Kutta methods, one must often implement complex numerical stabilization techniques to compensate for constraint divergence. In the discrete mechanics setting, this complexity is completely avoided. For more about variational integrators and their development see [15], [11], and [10].

[1]Here we have used the *slot derivative* notation where $D_i f(\cdot)$ represents a derivative of f with respect to its ith argument.

A. Kinematic Configuration Variables

For implementation convenience, when modeling the dynamics of mechanical systems, it is often desirable to reduce the full configuration space into a subset of configurations that follow kinematic paths. One way that this reduction can occur is by assuming that an actuator is capable of sufficiently controlling a given degree-of-freedom along any arbitrary path [8], [3]. Then the dynamics associated with the new "kinematic" configuration variables may be disregarded. This mixed kinematic-dynamic approach has several practical advantages.

First, the modeling of the dynamics of the actuators themselves is completely decoupled from the modeling of the dynamic system that is actually of interest. Second, from a practical perspective, in an experimental setup, it is often much easier to close the loop around velocities or positions than it is around forces. For example, no current sensing components or load cells are required in motor controllers, only encoders are used. This reduces both total component count, and system complexity. Additionally the motor control becomes much less dependent on having an accurate motor model.

In the continuous time setting this mixed kinematic-dynamic model [14] is generated by defining the system configuration q to have two parts a dynamic part q_d, and a kinematic portion ρ i.e. $q = [q_d \;\; \rho]^{\mathrm{T}}$. To ensure that the kinematic trajectory is twice differentiable in time we define the kinematic portion of the input u to be $\ddot{\rho}$. Then when the governing second order differential equation is converted into the standard form of $\dot{x} = f(x,u)$ trivial integrators ensure the kinematic trajectories are consistent.

In the discrete mechanics setting, the configurations at time t_k are once again split into a dynamic part q_k and a kinematic part ρ_k; where we have dropped the d subscript from the dynamic portion to avoid confusion with a time index specification. Next the DEL equations are separated into two separate sets of equations, one for the dynamic variables and the other for the kinematic variables. The kinematic variable assumption implies that at some (q_k, ρ_k) there will always exist some input to the system u_k, that will satisfy the kinematic DEL equations. Thus we simply drop this equation and set ρ_{k+1} to be a "kinematic input".

B. State Space Form

Here, temporarily assume that there are no kinematic configuration variables. In order to use classic tools from discrete system theory (such as linearizations, optimal linear time-varying control, etc.) we must be able to cast our problem into the standard nonlinear, first-order explicit difference equation

$$x_{k+1} = f_k(x_k, u_k). \tag{5}$$

This transformation can be done by first noticing that the second term in Eq. (4) is constant with respect to q_{k+1}.[2]

[2]Even though we illustrate this idea with the unforced, unconstrained DEL equations, analogous reasoning applies equally to systems with forcing and/or constraints.

Thus, define the following

$$p_k = D_2 L_d(q_{k-1}, q_k). \tag{6}$$

This new variable p_k is the discrete generalized momentum that is preserved exactly by the integrator.[3] Note that by defining p_k the explicit dependence on q_{k-1} has disappeared from the mapping provided by Eq. (4); it is now a one-step method $(q_k, p_k) \to (q_{k+1}, p_{k+1})$. As such we now define the state of the system and the governing difference equation as[4]

$$x_{k+1} = \begin{bmatrix} q_{k+1} \\ p_{k+1} \end{bmatrix} = f(x_k, u_k). \tag{7}$$

In this representation the mapping is now explicit, but in practice, one must still solve the implicit DEL given in Eq. (4). However, the key insight is that the existence of $f(x_k, u_k)$ is guaranteed by the Implicit Function Theorem (provided $D_2 D_1 L_d(q_k, q_{k+1}, t_k, t_{k+1})$ is non-singular at q_k, p_k, u_k).

Without going through the derivation of more complex forms of the DEL equations or demonstrating their analogous transformations to the explicit form of Eq. (7) we define the discrete system state for a general system in the present framework at a time $t = t_k$ as

$$x_k = \begin{bmatrix} q_k \\ \rho_k \\ p_k \\ v_k \end{bmatrix} \tag{8}$$

where v_k is defined as the finite-difference velocity of the kinematic configuration variables i.e. $v_k = \frac{q_k - q_{k-1}}{t_k - t_{k-1}}$. While this addition to the state is not strictly necessary, in the optimal control calculations discussed in Section III-B the v_k term is used in the cost function as a term to weight the "control effort" of the kinematic inputs.

Interestingly, unlike the continuous time setting where the state typically involves configurations and their velocities, the state here depends only on configurations. Even the generalized momentum terms p_k can be exactly calculated from configuration data at two consecutive time steps. This is beneficial for embedded system control. In a traditional discrete system representation the state has both configuration and velocity terms i.e. $x = [q \;\; \dot{q}]$. In a real-world setting, if one were to control this discrete system with full-state feedback one of two things would be necessary – i) noisy measurements of velocities, or ii) an estimator to fill in unobserved velocity states. This complexity is eliminated by the discrete mechanics strategy where only configurations are needed to completely fill in the state. Finally we define the new augmented input as $\bar{u}_k = [u_k \;\; \rho_{k+1}]$, and the augmented configuration as $\bar{q}_k = [q_k \;\; \rho_k]$. Note that the $k+1$ subscript on ρ in $\bar{u}$ is because at timestep t_k the kinematic input to the system is defined as the configuration of the kinematic variables at the next timestep, t_{k+1}.

[3]This conserved momentum will not be the exact momentum of the actual system but rather it is the exact momentum of a nearby *modified Lagrangian* system [13], [5]. In fact every variational integrator method has a conserved modified Hamiltonian that differs from the system's actual Hamiltonian by $O(h^p)$ where h is the timestep used in the integrator, and p is order of the variational integrator [4].

[4]Although we have not explicitly stated it, u is the input to the system and may consist of approximated time integrals of forcing terms as well as information about the trajectory of the kinematic configuration variables.

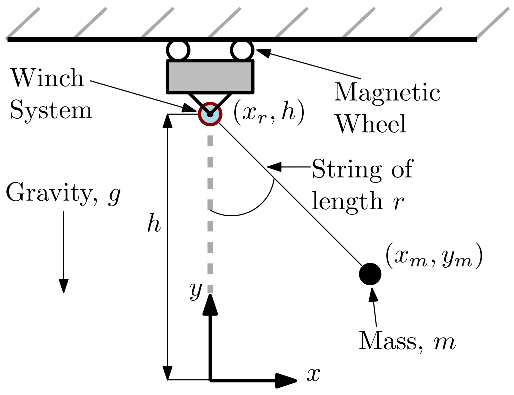


Fig. 1: Schematic of example system including relevant geometric parameters.

III. Implementation Details

In this section we discuss the details of the actual implementation of this discrete mechanics framework to an experimental system.

A. System Model

The example system in this work consists of a single magnetically-suspended robot driving in the plane with a mass hanging from an articulated string as shown in Fig. 1. There are two kinematic inputs to the system, the horizontal position of the robot x_r, and the length of the string r. Thus the augmented configuration and the augmented inputs are given by

$$\bar{q}_k = \begin{bmatrix} x_{m,k} \\ y_{m,k} \\ x_{r,k} \\ r_k \end{bmatrix} \quad \text{and} \quad \bar{u}_k = \begin{bmatrix} x_{r,k+1} \\ r_{k+1} \end{bmatrix}. \tag{9}$$

B. Software

The majority of the software implementation for simulations, and integrations is handled by `trep`, a Python module for simulating rigid body mechanical systems in generalized coordinates[5]. In `trep`, the user first defines an arbitrary mechanical system by specifying properties of transforms in $SE(3)$ between coordinate systems that follow a tree structure which scales well to high-degree-of-freedom systems [9]. The user can then add several types of constraints, potential fields, forces, and kinematic configuration variables to complete the system definition. The dynamics are then discretely integrated using a midpoint variational integrator [15], [10].

Beyond simple simulations, `trep` also has the ability to perform exact first and second derivatives of both the discrete and continuous dynamics, linearizations of the discrete

[5]Software is available for free download at `http://trep.googlecode.com` discussed in more detail in [9].

system about system trajectories, and a nonlinear projection-based optimization algorithm for trajectory synthesis. For the results in this work `trep` was used to generate optimal system trajectories based on arbitrary, possibly unfeasible system trajectories. Once an optimal, feasible trajectory was generated, `trep` was also used to determine a time-varying, stabilizing proportional controller for tracking the optimal trajectory and linearizations of the system for use in filtering algorithms.

The optimization algorithm is based on one originally presented in [6], and further explored in [14], [8], and [12]. One of the primary features of the original algorithm, as it was stated in [6], was the infinite-dimensional framework in which the algorithm was developed. This allowed the optimization to take place in the system's natural, continuous time representation wherein powerful adaptive time stepping integration schemes could be used to integrate all necessary differential equations. This makes the algorithm more robust to user-tuning as it removes a modeling decision that could significantly impact the results.

In [12], a modified discrete-time version of this algorithm was presented. Variational integrator's inherent stability and conservation properties facilitate the use of this algorithm in a discrete-time setting as the choice of timestep has a much smaller impact on the results of the optimization. In [12] the discrete algorithm was successfully used to solve several trajectory generation problems.

For completeness, a very brief overview of the algorithm is presented here. Given an initial condition $x(0) = x_0$ and a desired trajectory $\xi_d = (x_d(k), \bar{u}_d(k))$ over the discrete horizon $[k_0, k_f]$ where the pair $(x_d(k), \bar{u}_d(k))$ may or may not satisfy the system dynamics, solve the following constrained optimization problem:

$$\begin{aligned} &\text{minimize} && J(\xi) = \sum_{k=0}^{k_f-1} \ell(k, x(k), \bar{u}(k)) + m(x(k_f)) \qquad (10)\\ &\text{subject to} && x(k+1) = f(x(k), \bar{u}(k)), \quad x(0) = x_0 \end{aligned}$$

with $\xi = (x(k), \bar{u}(k))$. The summation argument and terminal cost are given by standard LQR terms

$$\begin{aligned} \ell(k, x(k), \bar{u}(k)) &= (x(k) - x_d(k))^{\mathrm{T}} Q\, (x(k) - x_d(k)) && \text{(11a)}\\ &\quad + (\bar{u}(k) - \bar{u}_d(k))^{\mathrm{T}} R\, (\bar{u}(k) - \bar{u}_d(k)) \\ m(x(t_f)) &= \left(x(k_f) - x_d(k_f)\right)^{\mathrm{T}} P_1 \left(x(k_f) - x_d(k_f)\right) && \text{(11b)} \end{aligned}$$

where Q, R, and P_1 are positive definite weighting matrices.

To solve this problem starting at iteration n, a descent direction ζ^n is determined and added to the current iteration ξ^n to produce $\eta^n = \xi^n + \zeta^n$. The resulting pair $\eta^n = (x^n(k), u^n(k))$ does not obey the system dynamics so we define a *projection operator* as a feedback law to project back onto the feasible set using the following:

$$\begin{aligned} x_0 &= x_0^n \\ x_{k+1} &= f(x_k, \bar{u}_k) \qquad (12)\\ \bar{u}_k &= \bar{u}_k^n - K_k(x_k - x_k^n) \end{aligned}$$

This yields the next feasible iteration ξ^n. The feedback gain in Eq. (12) is found by solving a finite-horizon discrete LQR problem using the discrete linearization about the current trajectory ξ^n [1]. Now, a new descent direction ζ^{n+1} can be found and added. The resulting infeasible pair η^{n+1} is projected, and the process continues until optimality conditions are satisfied.

C. Experimental Setup

The experimental system consists of a single inverted, magnetically-suspended robot driving in a plane and controlling a winch for string-length control. The robot uses only digital encoders for motor feedback, and the motor control loop is closed around the kinematic configuration variables x_r and r.

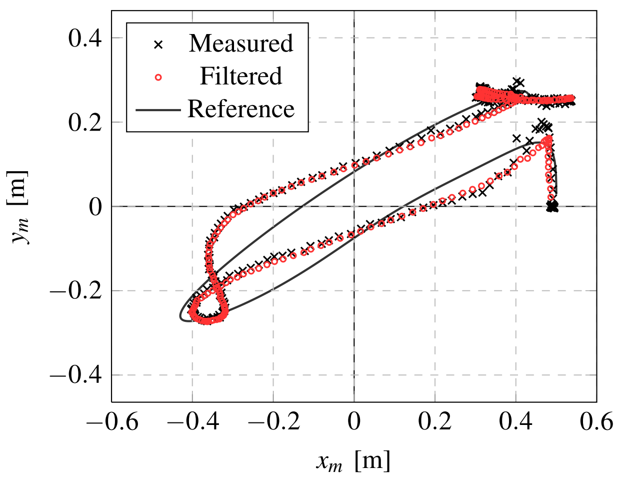
Optimizations are performed in `trep` *a priori* to obtain desired configuration paths for the kinematic variables x_r and r as well as the full-state feedback stabilizing controller $K(k)$. Experimental control is handled through the Robot Operating System (ROS), and full configuration feedback is obtained only through a single Microsoft Kinect®[6]. In an individual trial the result of a `trep` optimization is loaded into a ROS node, and using information about the desired initial configuration of the system a calibration is performed. Point cloud data from the Kinect is obtained at approximately 30 Hz, and this frequency drives the control frequency of the entire system. Knowing that the controller will run at 30 Hz, the optimization is performed assuming a time step of 1/30 s. Every time the ROS system receives a point cloud from the Kinect, a processing/ tracking node provides a measurement of the of the augmented configuration $\bar{q}$. This measurement is then filtered using a standard extended Kalman Filter (EKF) algorithm that uses `trep` to step a variational integrator for predicting the system state as well as provide linearizations of the system for the EKF update equations. This filtered estimate is then passed into the control law given in Eq. (12) to obtain the $\bar{u}_k$. The resulting command is then wirelessly sent to the robot, and it runs a high-frequency control loop around $\bar{u}_k$ until a new command is received.

IV. Results

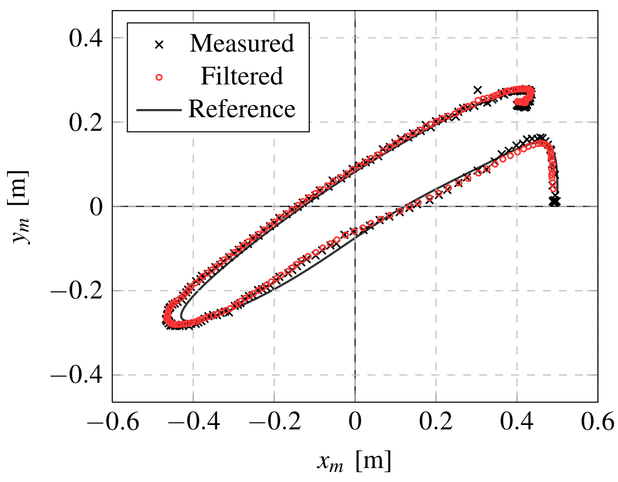
In this section we present both simulation-based and experimental results obtained by applying the discrete mechanics controller and estimator synthesis outlined in the previous sections to the sample system described in Section III. We highlight the benefits of the present framework in a practical setting.

One of the features of the discrete mechanics framework presented is the stability of the method over a large range of time steps. Thus the real-world frequency constraints of a particular system are easily accommodated. In other words, given a particular piece of experimental hardware with a constrained frequency e.g. the Kinect running at 30 Hz, the control system design and structure can be set to run in lockstep with the frequency of the sensor. Even in situations

[6]All ROS nodes for controlling the experimental system are available at `https://github.com/jarvisschultz/puppeteer_stack`



(a) Open-loop



(b) Closed-loop

Fig. 2: Parametric plots of the dynamic configuration variables for a individual trials of the experimental system. An optimal reference and a corresponding time-varying, proportional stabilizing controller have been determined using `trep`. The difference between the two plots is that in Fig. 2a the reference inputs to the system have been sent to the robot open-loop, and in Fig. 2b the stabilizing controller is utilized. Note that the last two seconds of the reference trajectory has $(x_m, y_m) \approx (0.43, .25)$ to show the controller's ability to stabilize to a set point. In Fig. 2a the mass swinging around this point is clearly seen.

where the sensor has a relatively low update frequency the stabilizing properties of the controller and the convergence rates of the estimator are likely to be much better than a continuous time controller that has been modified to run at the desired frequency through something like a zero-order hold transform.

Figure 3 shows simulated effects of the choice of system representation on performance of the EKF algorithm at a range of timesteps. Each point on the plot was generated by running 1000 trials of the filter on a nominal feasible trajectory for the system described in Fig. 1. In the "prediction" step of the EKF, actual samples from the nominal trajectory were used i.e. the predictions without additive noise corresponded exactly with the nominal trajectory. Measurements were simulated by adding Gaussian noise to the nominal trajectory. For each trial, the L_2 error between the filtered signal and the reference signal was determined at each timestep. These errors were then averaged to produce an "error norm" for each trial. The error norms were then averaged, and their standard deviations calculated to produce the points and error bars on Fig. 3. For the upper, solid curve the local linearization was performed by evaluating the infinitesimal derivatives of the continuous dynamics about the current best estimate, and then using an explicit Euler approximation to convert this into a discrete linearization. While this is arguably the simplest possible choice for a discrete approximation, it is often used in practice because of its simplicity. For the bottom curve, `trep` was used to provide the exact linearization of the discrete mechanics representation of the system. It is easily seen that as the filter frequency decreases, the performance of the discrete mechanics representation significantly outperforms the Euler

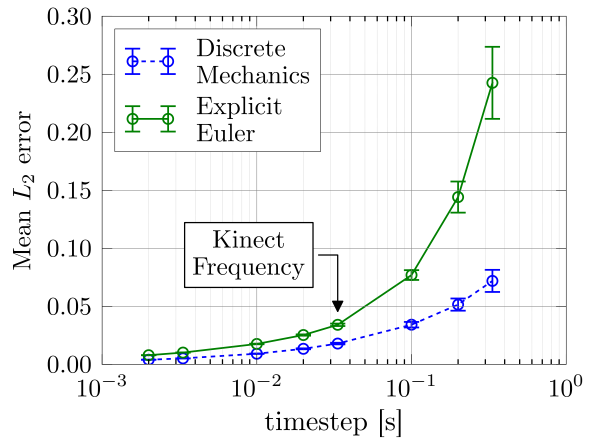


Fig. 3: Plot illustrating variations in EKF filter performance using two different discrete representations of the continuous dynamics. Both curves are simulated from 1000 trials, with Gaussian noise added to produce "measurements".

approximation. Also note that even at the 30 Hz frequency of the Kinect®, the mean L_2 error produced with the Euler approximation is nearly double the error of the discrete mechanics representation.

Figure 2 shows an experimental comparison between the open-loop and the closed-loop performance of the system. There are several characteristics to note. First, on the length scale associated with the reference trajectory the Kinect and associated processing algorithms provide a relatively noise-free measurement. However the large "jumps" seen occasionally are handled robustly by the filter. The second thing to note is that the open-loop trial does a quite poor

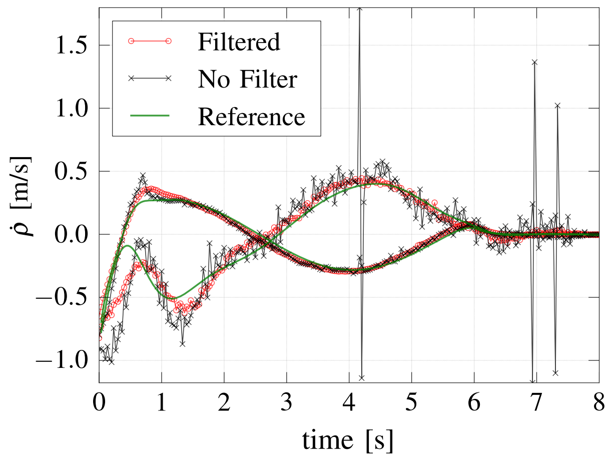


Fig. 4: Experimental demonstration of the effects the filtering algorithm has on the velocity commands sent to the robot.

job of tracking the reference trajectory. This is attributed to several factors: i) ROS is not actually real-time the mean *dt* for the trial in Fig. 2a is 0.0337 with a standard deviation of 0.004. ii) errors in the initial system configuration iii) errors the robot's motor controller iv) not having a perfect system model. The closed-loop tracking results show that the system works well. Especially important to note is that the actual embedded system is quite simple, nothing more than cheap DC motors, hobbyist electronics and the Kinect sensor. For controlling such an underactuated, nonlinear, and lowly damped system, one could imagine far more complex experimental setups.

Even though the Kinect data appears relatively smooth in Fig. 2 the EKF is performing an important role. This is illustrated in Fig. 4. The solid line is the finite-difference estimate of the derivative of the optimal reference for ρ. The other two curves are the actual commands sent to the robot during two different experimental trials one where the raw Kinect data has been used for feedback and one where the EKF algorithm has filtered the Kinect data. It is clearly seen that the filter yields a much smoother set of controls for the system.

V. Conclusions and Future Work

In this work we have presented a discrete mechanics based framework for modeling general mechanical systems that is useful when implementing control systems on an embedded system. The framework has desirable characteristics of a discrete approximation for use in an embedded system. Firstly, it is a one-step method that enables one to apply classical digital control tools. Even though it is only a one step method, the discrete integrator is guaranteed to perfectly conserve any system constraints, and has guarantees about energy behavior and Hamiltonian conservation. Several features of this framework were illustrated by looking at experimental and simulated results. While this scheme is slightly more complex that a simple one-step Euler integration scheme for approximating continuous systems, the guarantees about its numerical structure make it particularly appealing for modeling highly nonlinear systems, nearly-conservative systems, or long time horizon systems. Additionally, the effective dimension of the system state is the same as that of a one-step Euler method. Kinematic configuration variables provide a modeling strategy that facilitates control implementation on an embedded system, and they are easily incorporated into this framework.

In this work, we showed that this discrete mechanics framework yielded better filter performance on a classic EKF algorithm; this is almost exclusively due to the fact that the linearization of the discrete mechanics representation is more accurate than a linearization of an explicit Euler discrete approximation. Discrete mechanics with variational integrators also shows promise with sampling based filters such as a particle filter. Recent work has shown that variational integrators have equivalent conservation properties for stochastic systems as well [2]. Investigating the use of variational integrators for sampling-based filters in an experimental setting is the next logical progression for this work.

References

[1] B. D. O. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Dover Publications, Feb. 2007.

[2] N. Bou-Rabee and H. Owhadi, "Stochastic variational integrators," *IMA Journal of Numerical Analysis*, vol. 29, no. 2, pp. 421–443, Apr. 2009.

[3] F. Bullo and A. D. Lewis, *Geometric Control of Mechanical Systems*, ser. Texts in Applied Mathematics. New York-Heidelberg-Berlin: Springer Verlag, 2004, vol. 49.

[4] E. Hairer, C. Lubich, and G. Wanner, "Geometric numerical integration illustrated by the Störmer/Verlet method," *Acta Numerica*, vol. 12, pp. 399–450, 2003.

[5] ——, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer, Apr. 2006.

[6] J. Hauser, "A projection operator approach to the optimization of trajectory functionals," in *IFAC World Congress*, Barcelona, Spain, July 2002.

[7] J. P. Hespanha, *Linear Systems Theory*. Princeton, New Jersey: Princeton Press, sep 2009.

[8] E. Johnson and T. D. Murphey, "Dynamic modeling and motion planning for marionettes: Rigid bodies articulated by massless strings," in *International Conference on Robotics and Automation*, Roma, Italy, Apr. 2007, pp. 330–335.

[9] E. R. Johnson and T. D. Murphey, "Scalable variational integrators for constrained mechanical systems in generalized coordinates," *IEEE Transactions on Robotics*, vol. 25, pp. 1249–1261, Oct. 2009.

[10] O. Junge, J. E. Marsden, and S. Ober-blöbaum, "Discrete mechanics and optimal control," in *IFAC World Congress*, July 2005, p. 6.

[11] J. E. Marsden and M. West, "Discrete mechanics and variational integrators," *Acta Numerica*, 2001.

[12] T. Murphey and E. Johnson, "Control aesthetics in software architecture for robotic marionettes," in *American Control Conference (ACC), 2011*, July 2011, pp. 3825–3830.

[13] D. Pekarek and T. Murphey, "A backwards error analysis approach for simulation and control of nonsmooth mechanical systems," in *2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, Dec. 2011, pp. 6942–6949.

[14] J. Schultz and T. Murphey, "Trajectory generation for underactuated control of a suspended mass," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, May 2012, pp. 123–129.

[15] M. West, "Variational integrators," Ph.D. dissertation, California Institute of Technology, 2004.