Lie Group Integrators for Animation and Control of Vehicles

Marin Kobilarov Keenan Crane Mathieu Desbrun
California Institute of Technology

This paper is concerned with the animation and control of vehicles with complex dynamics such as helicopters, boats, and cars. Motivated by recent developments in discrete geometric mechanics we develop a general framework for integrating the dynamics of holonomic and nonholonomic vehicles by preserving their state-space geometry and motion invariants. We demonstrate that the resulting integration schemes are superior to standard methods in numerical robustness and efficiency, and can be applied to many types of vehicles. In addition, we show how to use this framework in an optimal control setting to automatically compute accurate and realistic motions for arbitrary user-specified constraints.

Categories and Subject Descriptors: I.3.7 [**Three-Dimensional Graphics**]: Animation;
I.6.8 [**Simulation and Modeling**]: Animation

1. INTRODUCTION

A *vehicle* is an actuated mechanical system that moves and interacts with its environment, such as a car, helicopter, or boat. While vehicles constitute a highly visible component of the world around us, the topic of vehicle dynamics has received little attention in the computer animation literature, and only a few off-the-shelf solutions for vehicle animation and control exist [Craft Animations ; Kineo CAM]. This deficiency is in sharp contrast with other animation and control tasks such as rigid/articulated/deformable body simulation, fluid phenomena, and character animation for which a plethora of techniques are available. Additionally, human familiarity with vehicles' highly idiosyncratic trajectories makes it difficult (or simply tedious) for artists to capture the essence of vehicle motion. Although locomotion and actuation have been thoroughly studied by roboticists [Latombe 1991; Murray et al. 1994], these tools have not been adequately adapted to computer graphics applications where performance, controllability, and ease of implementation are key.

At first glance, vehicle simulation appears simple: a vehicle is easily modeled by its *pose* in the world and a set of *internal variables* that describe its shape and/or internal dynamics. For example, a basic car simulator might operate on a pose (the car's current position and orientation) defined as $(x, y, \theta) \in SE(2)$ and two internal variables giving the orientation of the front wheels and the rolling angle of the rear wheels. Interaction with the environment could be described by either external forces such as contact forces, or by constraints on the velocities such as wheel rolling constraints arising from traction with the ground. However, this last constraint is not as simple as it may seem: it means that a car can only move in the direction in which the front wheels are pointing to. Such constraints on the velocities are called *nonholonomic* and are one of the reasons why parallel parking is a non-trivial task. Additionally, non-holonomic constraints, which restrict the possible *motions* of a mechanical system, are notoriously more delicate to enforce numerically [Cortés and Martínez 2001; Bloch 2003] than *holonomic* constraints, which restrict only the possible poses.

Contributions. This paper introduces general integrators for vehicles which handle both holonomic and non-holonomic constraints. We provide a self-contained description of generic numerical algorithms for vehicle integration, spell out specific integrators for computer animation in the case of a car, a helicopter, a boat, and a snake-board, and demonstrate numerical superiority compared to traditional Euler and Runge-Kutta integrators.

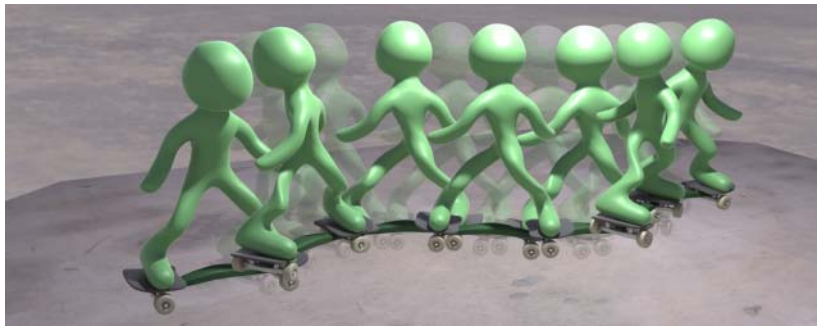


Fig. 1: *A snakeboard (see description in Fig. 7) is animated using our nonholonomic integrator that realistically and efficiently accounts for hip and foot motion. This paper presents a general framework for designing variational holonomic integrators and structure-respecting nonholonomic integrators for all sorts of vehicles, including cars, boats, and helicopters. These Lie group-based integrators are particularly robust for large time steps, and compete in efficiency with RK methods for small time steps.*

Our work extends the recently developed *geometric Lie group integrators* [Bou-Rabee and Marsden 2009] to provide a principled approach to the design of structure-preserving integrators for vehicles. Compared to previous methods (most notably Cortés [2002], Fedorov and Zenkov [2005], McLachlan and Perlmutter [2006] and de Leon et al. [2004]) our approach to nonholonomic systems with symmetries is more general as it can handle *arbitrary* group structure, constraints, and shape dynamics, and is not restricted to a configuration space that is either solely a group or has a Chaplygin-type symmetry. As a result, our formulation contains an additional discrete momentum equation analogous to the continuous case (e.g., as described in [Bloch et al. 1996]) that explicitly accounts for and respects the interaction between symmetries and constraints in the vehicle dynamics.

Our resulting numerical schemes provide several practical benefits directly relevant to computer graphics applications. First, a user can easily apply our framework to any vehicle by supplying its Lagrangian and constraints. Second, there is no need to use local coordinates that require expensive chart-switching, or special handling of singularities and numerical drift as required in previous methods. Additionally, fairly large time steps can be used without affecting numerical stability, making the method practical for the frame rates often used in animation. Finally, motion is computed in the minimum state-space dimension, thereby avoiding the computational burden that the conventional use of Lagrange multipliers induces. Consequently, our formulation allows the design of motions for systems with intricate dynamics through a simple algorithmic procedure, while benefiting from the desirable properties of discrete mechanics and Lie group methods such as robust and predictive numerics.

Outline. After a quick review of the current state of the art on regular and Lie group variational integrators in Sec. 2, we present a formal, general treatment of the discrete variational principles used to derive our integrators in Sec. 3. We then present the resulting integrators, first for purely holonomic systems (Section 4), then for nonholonomic systems (Section 5), along with the explicit expressions necessary for implementation. The specific cases of a car, a boat, a helicopter, and a snakeboard are detailed as concrete examples, though virtually any "exotic" vehicle could be derived from our general exposition. We finally point out in Sec. 6 how these integrators can be directly used in the context of optimal control to design specific trajectories while minimizing a cost function such as travel time or fuel consumption.

2. BACKGROUND ON TIME INTEGRATORS

A mechanical integrator advances a dynamical system forward in time. Such numerical algorithms are typically constructed by directly discretizing the differential equations that describe the trajectory of the system, resulting in an *update rule* to compute the next state in time. The integrators employed in this paper are instead based on the discretization of *geometric variational principles*. We start with a brief review of variational integrators, as well as their recent extensions that handle group structure and symmetries (e.g., update of rotation matrices).

2.1 Variational Integrators

Variational integrators [Marsden and West 2001] are based on the idea that the update rule for a discrete mechanical system (i.e., the time stepping scheme) should be derived *directly* from a variational principle rather than from the resulting differential equations. This concept of using a unifying principle from which the equations of motion follow (typically through the calculus of variations [Lanczos 1949]) has been favored for decades in physics. Chief among the variational principles of mechanics is *Hamilton's principle* which states that the path $q(t)$ (with endpoint $q(t_0)$ and $q(t_1)$) taken by a mechanical system extremizes the *action integral* $\int_{t_0}^{t_1} \mathcal{L}(q,\dot{q})dt$, i.e., the time integral of the *Lagrangian* $\mathcal{L}$ of the system, equal to the kinetic minus potential energy of the system. A number of properties of the Lagrangian have direct consequences on the mechanical system. For instance, a *symmetry* of the system (i.e., a transformation that preserves the Lagrangian) leads to a momentum preservation—see [Stern and Desbrun 2006] for a longer introductory exposition aimed at computer animation.

Although this variational approach may seem more mathematically motivated than numerically relevant, integrators that respect variational properties exhibit improved numerics and remedy many practical issues in physically based simulation and animation. First, variational integrators automatically preserve (linear and angular) momenta exactly (because of the invariance of the Lagrangian with respect to translation and rotation) while providing good energy conservation over exponentially long simulation times for non-dissipative systems. Second, arbitrarily accurate integrators can be obtained through a simple change of quadrature rules. Finally, they preserve the *symplectic structure* of the system, resulting in a much-improved treatment of damping that is essentially independent of time step [Kharevych et al. 2006]—a crucial property in computer graphics where coarse and fine simulations of the same system are often needed for preview purposes.

Practically speaking, variational integrators based on Hamilton's principle first approximate the time integral of the continuous Lagrangian by a *quadrature*, function of two consecutive states q_k and q_{k+1} (corresponding to time t_k and t_{k+1}, respectively):

$$L(q_k, q_{k+1}) \approx \int_{t_k}^{t_{k+1}} \mathcal{L}(q(t), \dot{q}(t))\mathrm{d}t.$$

Equipped with this "discrete Lagrangian," one can now formulate a discrete principle for the path $\{q_0, ..., q_N\}$ defined by the successive position at times $t_k = kh$. This discrete principle requires that

$$\delta \sum_{k=0}^{N-1} L(q_k, q_{k+1}) = 0,$$

where variations are taken with respect to each position q_k along the path. Thus, if we use D_i to denote the partial derivative w.r.t the i^{th} variable, we must have

$$D_2L(q_{k-1}, q_k) + D_1L(q_k, q_{k+1}) = 0$$

for every three consecutive positions q_{k-1}, q_k, q_{k+1} of the mechanical system. This equation thus *defines* an integration scheme which computes q_{k+1} using the two previous positions q_k and q_{k-1}.

Simple Example. Consider a continuous, typical Lagrangian of the form $\mathcal{L}(q,\dot{q}) = \frac{1}{2}\dot{q}^T\mathbb{M}\dot{q} - V(q)$ (V being a potential function) and define the discrete Lagrangian $L(q_k, q_{k+1}) = h\mathcal{L}\big(q_{k+\frac{1}{2}}, (q_{k+1} - q_k)/h\big)$, using the notation $q_{k+\frac{1}{2}} := (q_k + q_{k+1})/2$. The resulting update equation is:

$$\mathbb{M}\,\frac{q_{k+1} - 2q_k + q_{k-1}}{h^2} = -\frac{1}{2}(\nabla V(q_{k-\frac{1}{2}}) + \nabla V(q_{k+\frac{1}{2}})),$$

which is a discrete analog of Newton's law $\mathbb{M}\,\ddot{q} = -\nabla V(q)$. This example can be easily generalized by replacing $q_{k+1/2}$ by $q_{k+\alpha} = (1-\alpha)\,q_k + \alpha\,q_{k+1}$ as the quadrature point used to approximate the discrete Lagrangian, leading

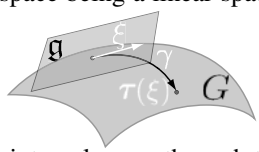
to variants of the update equation. For controlled (i.e., non conservative) systems, forces can be added using a discrete version of Lagrange-d'Alembert principle in a similar manner [Stern and Desbrun 2006].

2.2 Lie Group Integrators

Classical integrators, including the variational ones just described, advance a numerical solution in time by adding to the current configuration a displacement in $\mathbb{R}^N$. However, a number of systems have more complicated configuration spaces: a simple example most relevant to the remainder of this paper is that of rigid bodies, whose configuration space is the Lie group $SE(3)$, i.e., the group of Euclidean transformations; a member of this group is traditionally represented by a vector ($\in \mathbb{R}^3$) to encode translation and a rotation matrix ($\in SO(3)$) to encode orientation. This group, along with elements of its associated Lie algebra $\mathfrak{se}(3)$ (which can be thought as infinitesimal elements of $SE(3)$, i.e., instantaneous screw motions) and the exponential map, have already been proven useful in graphics [Alexa 2002; Kaufman et al. 2005]. Similarly, Lie group time integrators have been proposed in the mechanics literature to *automatically enforce that the updated poses remain within the proper group* without recourse to computationally-expensive reprojection or constraints [Hairer et al. 2006].

More abstractly, Lie group integrators preserve motion invariants and group structure for systems with a Lie group configuration space G. Its associated Lie algebra $\mathfrak{g}$ (respectively, Lie coalgebra $\mathfrak{g}^*$) is used to encode quantities such as generalized velocity and acceleration (respectively, generalized momentum and force). While a Lie group forms a smooth manifold, its associated algebras are simpler vector spaces—this latter fact makes the integration of velocity simple, even for curved configuration spaces. These special integrators often express the updated configuration in terms of a *group difference map* τ, i.e., a map that expresses changes in the group in terms of elements in its Lie algebra. The well-known *exponential map* was the first such map proposed for integration purposes in [Simo et al. 1992]. Retaining the Lie group structure and motion invariants under discretization has, since then, been proven to be not only a nice mathematical property, but also key to improved numerics, as they capture the right dynamics (even in long-time integration) and exhibit increased accuracy [Iserles et al. 2000; Bou-Rabee and Marsden 2009].

Throughout this paper we will use a generic configuration manifold $Q = M \times G$ where G is a Lie group (with Lie algebra $\mathfrak{g}$). In our case of vehicle dynamics, $G = SE(3)$ is typically the group of rigid body motions of an articulated body while M is a space of internal variables of the vehicles. Note that now, a vehicle's state is entirely defined by a point $q \in Q$, and its velocity $\dot{q} \in T_qQ$ (T_qQ being the tangent space of Q at q). The idea of Lie group integrators is to transform the equations of motion from the original state space TQ into equations on the *reduced space* $TM \times \mathfrak{g}$—elements of TG are translated to the origin and expressed in the algebra $\mathfrak{g}$. This reduced space being a linear space, standard integration methods can then be used as mentioned earlier.



The inverse of the transformation τ is used to map curves in the algebra back onto the group. Two standard group difference maps τ have been commonly used to achieve this transformation for any Lie group G:

- The exponential map $\exp : \mathfrak{g} \to G$, defined by $\exp(\xi) = \gamma(1)$, with $\gamma : \mathbb{R} \to G$ is the integral curve through the identity of the vector field associated with $\xi \in \mathfrak{g}$ (hence, with $\dot{\gamma}(0) = \xi$);
- Canonical coordinates of the second kind $\mathrm{ccsk} : \mathfrak{g} \to G$, $\mathrm{ccsk}(\xi) = \exp(\xi^1 e_1) \cdot \exp(\xi^2 e_2) \cdot \ldots \cdot \exp(\xi^n e_n)$, where $\{e_i\}$ is the Lie algebra basis.

A third choice for τ, valid only for certain matrix groups [Celledoni and Owren 2003] (which include the rigid motion groups $SO(3)$, $SE(2)$, and $SE(3)$), is the *Cayley map*:

$$\mathrm{cay} : \mathfrak{g} \to G,\ \mathrm{cay}(\xi) = (e - \xi/2)^{-1}(e + \xi/2).$$

Although this last map provides only an approximation to the integral curve defined by $\exp$, we include it because it is very easy to compute and thus results in a more efficient implementation. Other approaches are also possible, e.g., using retraction and other commutator-free methods; we will however limit our exposition to the three aforementioned maps in the formulation of the discrete reduced principle presented in the next section.

2.3 Notation

Table I summarizes variables used in the remainder of this paper, along with their usual computational representation. Most of these variables are typical quantities in rigid body simulation, except for r which encodes the *shape variables* (e.g., tire orientation for a car), and u which encodes the *control inputs* (e.g., torque on a car's steering wheel).

Symbol	Physical Meaning	Numerical Representation
x	translational position	vector $\in \mathbb{R}^3$
R	orientation matrix	matrix $\in \mathbb{R}^{3\times 3}$
v	linear velocity	vector $\in \mathbb{R}^3$
ω	angular velocity	vector $\in \mathbb{R}^3$
g	configuration	(x, R)
ξ	velocity	vector $\in \mathbb{R}^6$ $(= (v, \omega))$
μ	momentum	vector $\in \mathbb{R}^6$
f	external/control forces	vector $\in \mathbb{R}^6$
r	shape coordinates	vector $\in \mathbb{R}^{\#\text{ of shape variables}}$
u	control inputs	vector $\in \mathbb{R}^{\#\text{ of inputs}}$

Table I: *Common physical variables used in this vehicle simulation paper.*

3. DISCRETE REDUCED VARIATIONAL PRINCIPLES

The discrete equations of motion for the vehicles we consider are derived through a discrete reduced *d'Alembert-Pontryagin* variational principle, an extension of the Hamilton-Pontryagin principle proposed for graphics in [Kharevych et al. 2006]. This principle applies to systems with symmetries, holonomic and nonholonomic constraints, as well as forcing. The holonomic version was introduced in [Bou-Rabee and Marsden 2009] and extended to nonholonomic systems with symmetries in [Kobilarov 2007]. The nonholonomic integrators proposed in this latter reference apply to a general class of systems that were studied in [Bloch et al. 1996; Koon and Marsden 1997] and later in [Cendra et al. 2001a; 2001b]. This section provides a formal, general treatment of these discrete variational principles that we will use to derive our integrators. Note that in practice, the Lie groups relevant to vehicles are either $SE(2)$ or $SE(3)$; therefore, implementation-oriented readers can safely skip the formal exposition in this section and refer directly to the specific vehicle equations of motion in Sec. 4.

3.1 Holonomic Systems

Our discrete variational principle for *holonomic* systems follows [Bou-Rabee and Marsden 2009], with the addition of potential functions and forcing (i.e., controls, external forces, and damping). We consider systems that evolve on a finite dimensional Lie group G whose state is described by configuration $g \in G$ and body-fixed velocity $\xi \in \mathfrak{g}$, where $\mathfrak{g}$ is the Lie algebra of G. The momentum of the system is denoted $\mu \in \mathfrak{g}^*$ where $\mathfrak{g}^*$ is the Lie coalgebra of G. The system is subject to a body-fixed control force $f : [0, T] \to \mathfrak{g}^*$. In the discrete setting the continuous curves $g(t), \xi(t), \mu(t), f(t)$, for $t \in [0, T]$ are approximated by a discrete set of points at equally spaced time intervals. For example, $g : [0, T] \to G$ is given the temporal discretization $g^d = \{g_0, g_1, \ldots, g_N\}$ with $g_k := g(kh)$, where $h = T/N$ is the time step.

Discrete Variational Principle. The ingredients necessary to formulate the principle are the Lagrangian $\ell : G \times \mathfrak{g} \to \mathbb{R}$ and the group difference map $\tau : \mathfrak{g} \to G$. Similar to the example reviewed in Sec. 2.1, we use a simple symplectic Euler scheme. For holonomic vehicles on Lie groups the variational principle states that

$$\delta \sum_{k=0}^{N-1} h \left[\ell(g_{k+\alpha}, \xi_k) + \left\langle \mu_k, \tau^{-1}(g_k^{-1} g_{k+1})/h - \xi_k \right\rangle \right] + \sum_{k=0}^{N-1} h \langle f_{k+\alpha}, g_{k+\alpha}^{-1} \delta g_{k+\alpha} \rangle = 0 \tag{1}$$

where $\langle ., . \rangle$ is the natural pairing (for implementation purposes, this pairing will simply become a regular dot product since velocities and momenta are numerically represented through vectors). This principle can be regarded as the

extremization of a discrete action integral (as described in Sec. 2.1), to which we added a Lie group update constraint (via the multiplier $\mu_k \in \mathfrak{g}^*$), as well as the virtual work done by external forces (expressed by the latter sum). The update constraint states that the difference between two group poses g_k and g_{k+1} must be expressed using a Lie algebra element $\xi_k \in \mathfrak{g}$, which also plays the role of the average body-fixed velocity along that segment. The value $\alpha \in [0,1]$ again determines the quadrature point along each segment at which the Lagrangian and the control forces are evaluated, i.e., $g_{k+\alpha} = g_k \tau(\alpha h \xi_k)$ and $f_{k+\alpha} = (1-\alpha) f_k + \alpha f_{k+1}$. Typically, one picks $\alpha = 0$, $1/2$, or 1; in general the resulting algorithms are implicit, with $\alpha = 1/2$ yielding more accurate but more complex equations, while for certain simpler groups such as $G = SE(2)$ choosing $\alpha = 1$ provides an explicit integrator.

Discrete Equations of Motion. For the sake of simplicity, assume the Lagrangian consists only of a kinetic energy term, $\ell(g,\xi) = \langle \mathbb{I}\,\xi, \xi \rangle$, where $\mathbb{I}$ is the inertia tensor (including both mass and moments of inertia), and take $\alpha = 1$. Applying the variations in (1) directly results in the following equations

$$\mu_k = \mathbb{I}\,\xi_k, \tag{2}$$

$$g_{k+1} = g_k \tau(h\xi_k), \tag{3}$$

$$(\mathrm{d}\tau^{-1}_{h\xi_k})^* \mu_k - (\mathrm{d}\tau^{-1}_{-h\xi_{k-1}})^* \mu_{k-1} = h f_k. \tag{4}$$

Equation (2) means that the multiplier μ_k used to enforce the constraint in (1) can also be regarded as the *momentum* of the system [Kharevych et al. 2006]. Equation (4) represents the balance of momentum. The map $\mathrm{d}\tau_\xi : \mathfrak{g} \to \mathfrak{g}$ is called the *right-trivialized tangent* and is defined by

$$\mathrm{d}\tau_\xi\,\delta = (D\tau(\xi)\cdot\delta)\,\tau(\xi)^{-1},$$

where D is the directional derivative along δ. More intuitively, the derivative of the flow map τ at point $\tau(\xi)$ is taken in the direction δ, and the resulting vector $D\tau(\xi)\cdot\delta$ is translated by right multiplication with $\tau(\xi)^{-1}$ to yield the Lie algebra element $\mathrm{d}\tau_\xi\,\delta$—which can be thought of as the velocity expressed in a reference frame attached at the group origin. The map $\mathrm{d}\tau_\xi^{-1} : \mathfrak{g} \to \mathfrak{g}$ is the inverse, i.e. $\mathrm{d}\tau_\xi^{-1} \cdot \mathrm{d}\tau_\xi\,\delta = \delta$, and $(\mathrm{d}\tau_\xi^{-1})^* : \mathfrak{g} \to \mathfrak{g}$ is the inverse transpose defined by $\langle (\mathrm{d}\tau_\xi^{-1})^* \mu, \delta \rangle = \langle \mu, \mathrm{d}\tau_\xi^{-1}\,\delta \rangle$. In their general form these maps operate on Lie algebra elements; but in our context, as detailed in Sec. 4, they are easily implemented as matrices and the transpose operator becomes the simple matrix transpose. By thinking of these linear maps as change-of-basis operators, the equations of motion have a natural geometric interpretation sketched in Fig. 2. In Sec. 4 we will provide the detailed form of Eqs. (2)-(4) when $G = SE(3)$, $\alpha = 1$, and for both cases of $\tau = \exp$ or $\tau = \mathrm{cay}$, as these equations are especially well suited for animation purposes.

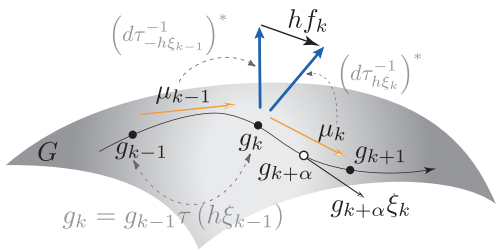


Fig. 2: *The discrete momentum equation* (4) *corresponds to a balance of projected momenta. Indeed, for any group pose $g_k \in G$, μ_{k-1} and μ_k are respectively the left-sided and right-sided momentum. In order to properly take their difference, they must first be transformed using the tangent maps to the same basis (one can think of this basis as being associated to point g_k). In this common basis, the difference between the two transformed vectors represents the* balance of momentum *at g_k. Consequently, it must equal the external force hf_k applied at that point.*

3.2 Nonholonomic Systems

In order to capture a wider variety of vehicle models we consider a more general class of systems that have internal variables and are subject to nonholonomic constraints. Such systems are defined on a manifold $Q = M \times G$, where M is the *shape space*. The group component G describes the pose of the vehicle (e.g., position and orientation for a car-like vehicle), just as in the holonomic case. The shape space M describes internal variables (such as tire orientation), and necessitates one additional ingredient known as the *connection*. The connection describes how changes in shape affect changes in the group, that is, it describes the nonholonomic constraints induced by internal variables. For example, the connection might encode how the orientation of the front wheels and the rolling of the rear wheels affect

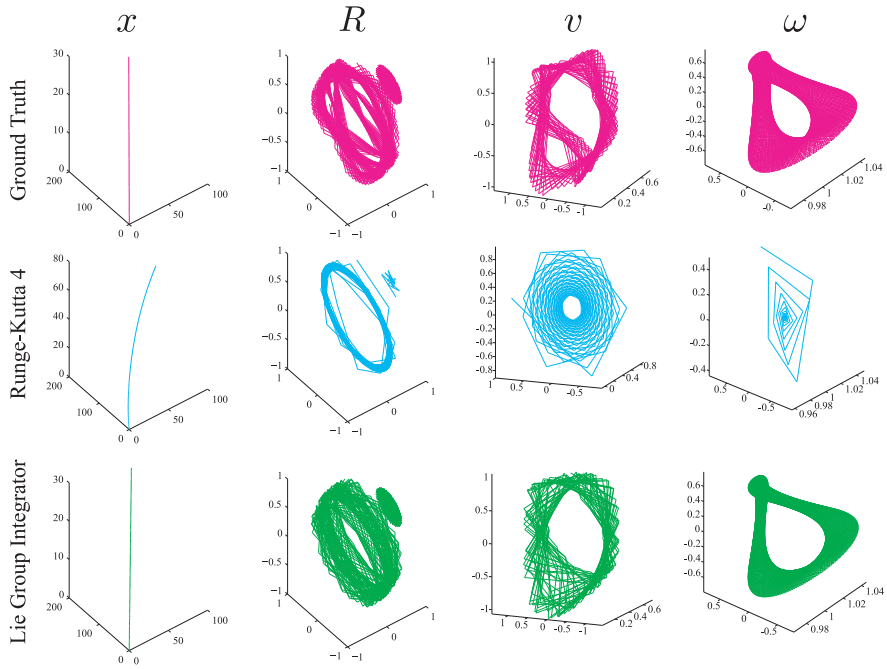


Fig. 3: *Position (x), orientation (R), linear velocity (v) and angular velocity (ω) versus time for a rigid body. While fourth-order Runge-Kutta fails for a large time step, our approach (bottom) reproduces the ground-truth orbits. This behavior is typical of structure-preserving integrators.*

the overall motion of the car. It also determines whether there is any continuing motion in the group when the shape is not changing or locked (usually associated with symmetries and conservation laws).

Nonholonomic Constraints. A nonholonomic constraint is described by a distribution $\mathcal{D}$ which is a collection of linear subspaces $\mathcal{D}_q \subset T_qQ$ specifying the permissible velocities at each point $q \in Q$. The *group orbit* of a point $q \in Q$ is the submanifold $\mathrm{Orb}(q) := \{gq \mid g \in G\}$, i.e., the set of points resulting from applying transforming to q with respect to a global basis fixed at the group origin. The tangent space $T_q \mathrm{Orb}(q)$ is called the *vertical space* at q and represents the set of velocities that leave the Lagrangian of the system invariant (i.e., the "symmetries" of the mechanical system mentioned earlier). The *intersection space* $\mathcal{S}_q \subset T_qQ$ is then the subspace of tangent vectors that satisfy the constraint while still preserving the Lagrangian, given by $\mathcal{S}_q = \mathcal{D}_q \cap T_q \mathrm{Orb}(q)$. Finally, define the vector space $\mathfrak{s}_q \subset T_qQ/G$ to be the set of Lie algebra elements that generate $\mathcal{S}_q$. In other words, curves in $\mathfrak{g}$ with velocities in $\mathfrak{s}_q$ become curves in $\mathcal{D}_q$ when translated to q. When deriving the equations of motion for nonholonomic vehicles, this subspace of the Lie algebra is associated with conserved quantities.

Dynamics. We will express vehicle trajectories in coordinates $(r, g) \in M \times G$ and use a basis e_a for the Lie algebra $\mathfrak{g}$ (hence an element $\xi \in \mathfrak{g}$ is written $\xi^a e_a$). Each system is subject to a control force $f : [0, T] \to T^*M$ which we assume is restricted to shape space. The Lagrangian $\ell : T_rM \times \mathfrak{g} \to \mathbb{R}$ is now a function of a velocity vector $\dot{r} \in T_rM$ in the shape space as well as the body-fixed velocity $\xi \in \mathfrak{g}$. For simplicity, we assume that the Lagrangian does not depend on the group configuration $g \in G$; this is often the case in practice since interaction with the environment is achieved directly through kinematic constraints rather than through a position-dependent potential.

Nonholonomic constraints and symmetry directions are encoded with a *nonholonomic connection* $\mathcal{A}$. The connection is a map, dependent on the shape coordinates r, that is linear in the velocities, leaves vertical vectors unchanged, and annihilates (i.e., maps to zero) vectors that satisfy the constraints while not lying in the symmetry directions, i.e. *horizontal* vectors. The connection is typically constructed as $\mathcal{A} = \mathcal{A}^{kin} + \mathcal{A}^{sym}$, where $\mathcal{A}^{kin}$ is the kinematic connec-

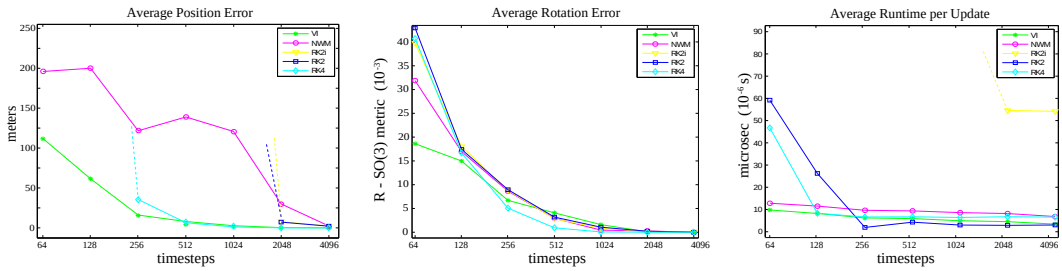


Fig. 4: *Stability and efficiency of our variational integrator for a rigid body: averaged over 20 runs using a wide range of initial conditions on a 4-minute animation, our integrator shows much improved robustness in the presence of large timesteps over RK2, implicit RK2 (RK2i), and RK4 (dashes indicate blow-ups), while its accuracy remains between 2nd and 4th order. Computational efficiency is similar to RKs for average time steps, but superior for small time steps as the solver then requires a single iteration for convergence; Newmark (NWM) leads to larger errors in position, as well as in rotation for large time steps.*

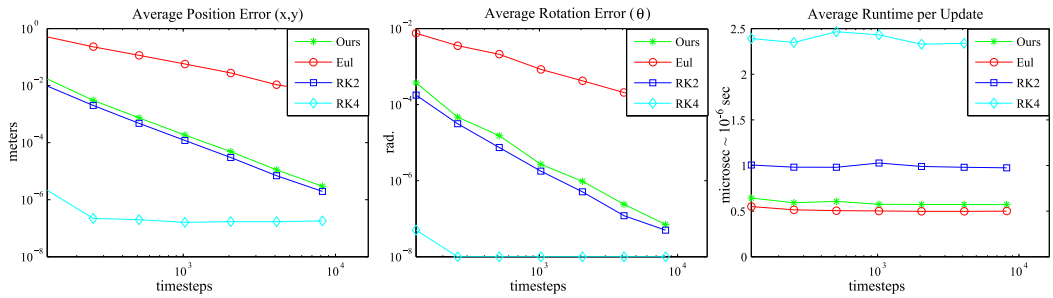


Fig. 5: *Stability and efficiency of our nonholonomic integrator for car trajectories: averaged over 50 runs using a large range of initial conditions and steering commands for a one-minute long animation, our nonholonomic integrator remains as accurate as RK2, at a fraction of the computational complexity.*

tion enforcing nonholonomic constraints and $\mathcal{A}^{sym}$ is the connection corresponding to symmetries in the constrained directions (i.e., the group orbit directions satisfying the constraints). These maps satisfy

$$\begin{aligned} \mathcal{A}^{\text{kin}}(q) \cdot \dot{q} &= 0, \\ \mathcal{A}^{\text{sym}}(q) \cdot \dot{q} &= \text{Ad}_g \, \Omega, \end{aligned} \tag{5}$$

where $\Omega \in \mathfrak{s}_r$, called the "locked angular velocity"(i.e., the body velocity obtained by locking the joints r; see [Bloch et al. 1996]), is computed from conservation laws along the constrained symmetry directions in $\mathcal{S}_q$. Intuitively, when the joints stop moving the system continues its motion uniformly along a curve (with tangent vectors in $\mathcal{S}$) with body-fixed velocity Ω and a corresponding spatial momentum that is conserved. By definition the principle connection can be expressed as

$$\mathcal{A}(q) \cdot \dot{q} = \text{Ad}_g(g^{-1}\dot{g} + \mathcal{A}(r)\dot{r}),$$

where $\mathcal{A}(r)$ is the local form and the two components in (5) can be added to get

$$g^{-1}\dot{g} + \mathcal{A}(r)\dot{r} = \Omega.$$

There are a number of systems for which such connection is known explicitly; in Sec. 5 we give two concrete examples of wheeled vehicles where this connection can be used directly.

Discrete Variational Principle. The discrete reduced d'Alembert-Pontryagin principle for nonholonomic systems is now defined as (using the notation $\xi_k := \Omega_k - \mathcal{A}(r_{k+\alpha}) \cdot u_k$):

$$\delta \sum_{k=0}^{N-1} h \left[\ell(r_{k+\alpha}, u_k, \xi_k) + \left\langle p_k, \frac{r_{k+1} - r_k}{h} - u_k \right\rangle + \langle \mu_k, \frac{\tau^{-1}(g_k^{-1} g_{k+1})}{h} - \xi_k \rangle \right] + \sum_{k=0}^{N-1} h \langle f_{k+\alpha}, \delta r_{k+\alpha} \rangle = 0,$$

$$\begin{aligned} \text{subject to:} \quad & \textit{vertical variations} \quad (\delta r_k, g_k^{-1} \delta g_k) = (0, \eta_k), \eta_k \in \mathfrak{s}_{r_k} \\ & \textit{horizontal variations} \quad (\delta r_k, g_k^{-1} \delta g_k) = (\delta r_k, -\mathcal{A}(r_k) \delta r_k), \end{aligned} \tag{6}$$

where $p_k \in T^*M$ is a Lagrange multiplier. In the above formulation, the variations δu_k, $\delta \Omega_k$, δp_k, $\delta \mu_k$ are unconstrained. The general nonholonomic integrator that results from this principle will be provided in Sec. 5.

3.3 Numerical Tests and Comparisons

The discrete variational approach yields an update scheme different from standard integration algorithms such as classical Runge-Kutta. We simulated a holonomic and a nonholonomic system representative of each class of vehicles in order to confirm the numerical advantages of this scheme. While long-time energy and structure preservation are the main performance benchmarks in the discrete mechanics literature [Marsden and West 2001; Cortés 2002; Kharevych et al. 2006; Bou-Rabee and Marsden 2009], animation quality also depends heavily on the *pose error* with respect to the ground truth trajectory of the vehicle. Also important for computer animation is the robustness of integrators when using large time steps. The results discussed next are averaged over 20 different runs with randomly chosen initial conditions and inertial properties.

Holonomic Rigid Body. Our first test uses four-minute trajectories of a simple, free-floating rigid body in $SE(3)$. Fig. 4 shows the resulting pose error and computation time for our method along with second and fourth order Runge-Kutta methods (*RK2*, implicit *RK2*, and *RK2* respectively) and Newmark as a function of the total number of time steps. In order to avoid singularities and costly switching of coordinate charts the *RK* methods were implemented using quaternions to represent rotations and reprojection (normalization of the quaternion) after each internal and external *RK* step. A standard form of implicit *RK2* was used (see, e.g., Sec.11.5 in [Dormand 1996]) to update the dynamics only while the reconstruction was performed explicitly using quaternions. The alternative of applying implicit RK2 to both the dynamics and reconstruction is not straightforward since it results in integration of nonlinearly constrained ordinary differential equations and requires a more complex approach. In order to implement Newmark's method for Lie groups we extended the Lie-Newmark method for rotational dynamics proposed by [Krysl and Endres 2005] to general Lie groups and applied it to the rigid body case. The graphs confirm the superior stability of the variational integrator, which behaves well even for time steps where *RK* methods blow up (dashed lines indicate NaN values or do not fit in the plot). Improved stability is likely due to the structure- and energy-preserving properties of our integrators as evidenced by Fig. 3 which shows the velocity and pose orbits of a particular run using a large time step. Ground truth in this figure was computed by running *RK4* with small time steps and subsampling the resulting trajectory. Fig. 4 also confirms that the improved behavior of the variational method does not come at a high cost. In fact, while not quite as accurate due to our choice of low-order quadrature, our method requires *less* computation time than *RK4* as the time-step gets smaller and is nearly twice as efficient as *RK4* on average. Interestingly, error accumulates faster in translational components than in rotational components. This could be explained by the fact that rotation is decoupled from translation and hence has its own error dynamics, whereas translation suffers from small cumulative errors in rotation. Finally, notice that the extensions of Newmark and implicit *RK2* schemes to Lie groups suffer, respectively, from significant position errors (in case of Newmark due to the different reconstruction equations) and lack of robustness in the presence of large time steps (in case of *RK2* due to difficulty in obtaining a good solution to the multiple implicit quadratic equations necessary to update the dynamics), rendering them both inadequate for graphics purposes.

Nonholonomic Car. Our numerical comparisons for nonholonomic motions use one-minute trajectories of a car with simple dynamics (Sec. 5.2.1). The vehicle is controlled using sinusoidal inputs of frequency and amplitude designed to produce nontrivial paths such as parallel parking, sharp turns, and winding maneuvers. Since the trajectory is relatively short the *RK* methods (Fig. 5) remain stable, due in part to the simpler group structure of $SE(2)$. Yet our integrator performs as well as *RK2* at *half* the computational time due to its explicit update scheme.

4. HOLONOMIC VEHICLES

In this section we consider vehicles that can be modeled as a single unconstrained underactuated rigid body. We allow the vehicle to have actuated parts whose motion does not affect the inertial properties of the system such as rudders, fins, or thrusters. The configuration space of the body is $G = SE(3)$ and the space of joint angles of the actuators is M. The dynamics of the system evolve on G only and the joint angles $\gamma \in M$ affect only the way control forces are applied. Such systems can be studied using the variational principle introduced in Sec. 3.1.

The vehicle *body-fixed* angular and linear velocities are denoted $\omega \in \mathbb{R}^3$ and $v \in \mathbb{R}^3$. The vector $(\omega, v) \in \mathbb{R}^6$ corresponds to a Lie algebra element $\xi \in \mathfrak{se}(3)$ through

$$\xi = \begin{bmatrix} \widehat{\omega} & v \\ \mathbf{0} & 0 \end{bmatrix}, \tag{7}$$

where the map $\widehat{\cdot} : \mathbb{R}^3 \to \mathfrak{so}(3)$ is defined by

$$\widehat{\omega} = \begin{bmatrix} 0 & -\omega^3 & \omega^2 \\ \omega^3 & 0 & -\omega^1 \\ -\omega^2 & \omega^1 & 0 \end{bmatrix}$$

The Lagrangian $\ell : \mathbb{R}^6 \to \mathbb{R}$ of such systems is given by

$$\ell(\omega, v) = \frac{1}{2} \left(\omega^T \, \mathbb{J} \, \omega + v^T \, \mathbb{M} \, v \right)$$

where $\mathbb{J}$ and $\mathbb{M}$ are the rotational and translational inertia matrices. The system is controlled using an input $u : [0, T] \to \mathbb{U}$, where $\mathbb{U} \subset \mathbb{R}^c$ ($c \leq n$) is the set of controls. The controls are applied with respect to a body-fixed basis defined by the columns of the matrix $F : M \to L(\mathbb{R}^6, \mathbb{R}^c)$ that depend on the actuator angles $\gamma \in M$ ($L(\mathbb{R}^m, \mathbb{R}^n)$ denoting a m-by-n real matrix). The total resulting control force is $F(\gamma)u$. In addition, the vehicle can be subject to external forces that depend on its current position and velocity defined by $f_{\text{ext}} : (SO(3) \times \mathbb{R}^3) \times \mathbb{R}^6 \to \mathbb{R}^6$. Such forces typically include gravity, buoyancy, drag, lift, or effects from wind and currents.

4.1 Continuous Equations of Motion

The continuous equations of motion (encoding the evolution of the position x, orientation R, and velocities ω and v) are derived from the reduced Lagrange-d'Alembert principle [Marsden and Ratiu 1999; Bullo and Lewis 2004] and have the standard form:

$$\begin{bmatrix} \dot{R} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} R \, \widehat{\omega} \\ R \, v \end{bmatrix}, \tag{8}$$

$$\begin{bmatrix} \mathbb{J} \, \dot{\omega} \\ \mathbb{M} \, \dot{v} \end{bmatrix} = \begin{bmatrix} \mathbb{J} \, \omega \times \omega + \mathbb{M} \, v \times v \\ \mathbb{M} \, v \times \omega \end{bmatrix} + F(\gamma)u + f_{\text{ext}}. \tag{9}$$

Eq. (9) are called the forced *Euler-Poincaré equations* and (8) are known as the *reconstruction equations*.

4.2 Variational Integrator

The discrete equations of motion to update position (x_k), orientation matrix (R_k), and velocities (ω_k and v_k) are derived through the variational principle (1). Choosing $\alpha = 1$ results in a symplectic Euler variational integrator (see [Kobilarov 2007; Bou-Rabee and Marsden 2009] for details) of the form:

$$\begin{bmatrix} R_{k+1} & x_{k+1} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} R_k & x_k \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \tau(h\omega_k) & h\,\mathrm{B}_\tau(h\omega_k)v_k \\ \mathbf{0} & 1 \end{bmatrix}, \tag{10}$$

$$\begin{aligned} &\mathrm{C}_\tau^T(h(\omega_k, v_k)) \begin{bmatrix} \mathbb{J}\,\omega_k \\ \mathbb{M}\,v_k \end{bmatrix} - \mathrm{C}_\tau^T(-h(\omega_{k-1}, v_{k-1})) \begin{bmatrix} \mathbb{J}\,\omega_{k-1} \\ \mathbb{M}\,v_{k-1} \end{bmatrix} \\ &= hF(\gamma_k)u_k + hf_{\text{ext}}((R_k, x_k), (\omega_{k-1}, v_{k-1})), \end{aligned} \tag{11}$$

for $k = 1, ..., N-1$. In these expressions, the map τ can be either the exponential map or the Cayley map (we denote by $\mathbf{I}_k$ the identity matrix of dimension k):

$$\exp(\omega) = \begin{cases} \mathbf{I}_3, & \text{if } \omega = 0 \\ \mathbf{I}_3 + \frac{\sin\|\omega\|}{\|\omega\|}\widehat{\omega} + \frac{1-\cos\|\omega\|}{\|\omega\|^2}\widehat{\omega}^2, & \text{if } \omega \neq 0 \end{cases}, \tag{12}$$

$$\mathrm{cay}(\omega) = \mathbf{I}_3 + \frac{4}{4 + \|\omega\|^2}\left(\widehat{\omega} + \frac{\widehat{\omega}^2}{2}\right) \tag{13}$$

The associated map $\mathrm{B}_\tau : \mathbb{R}^3 \to L(\mathbb{R}^3, \mathbb{R}^3)$ is, depending on the choice of τ, respectively:

$$\begin{aligned} \mathrm{B}_{\exp}(\omega) &= \begin{cases} \mathbf{I}_3, & \text{if } \omega = 0 \\ \mathbf{I}_3 + \left(\frac{1-\cos\|\omega\|}{\|\omega\|}\right)\frac{\widehat{\omega}}{\|\omega\|} + \left(1 - \frac{\sin\|\omega\|}{\|\omega\|}\right)\frac{\widehat{\omega}^2}{\|\omega\|^2}, & \text{if } \omega \neq 0 \end{cases} \\ \mathrm{B}_{\mathrm{cay}}(\omega) &= \frac{2}{4 + \|\omega\|^2}\left(2\mathbf{I}_3 + \widehat{\omega}\right) \end{aligned} \tag{14}$$

Finally, the map $\mathrm{C}_\tau : \mathbb{R}^6 \to L(\mathbb{R}^6, \mathbb{R}^6)$ is, depending on the choice of the group difference map τ:

$$\begin{aligned} \mathrm{C}_{\exp}((\omega, v)) &= \mathbf{I}_6 - \frac{1}{2}[\mathrm{ad}_{(\omega,v)}] + \frac{1}{12}[\mathrm{ad}_{(\omega,v)}]^2, \text{ or} \\ \mathrm{C}_{\mathrm{cay}}((\omega, v)) &= \begin{bmatrix} \mathbf{I}_3 - \frac{1}{2}\widehat{\omega} + \frac{1}{4}\omega\omega^T & \mathbf{0}_3 \\ -\frac{1}{2}\left(\mathbf{I}_3 - \frac{1}{2}\widehat{\omega}\right)\widehat{v} & \mathbf{I}_3 - \frac{1}{2}\widehat{\omega} \end{bmatrix}, \end{aligned}$$

$$\text{where:} \qquad [\mathrm{ad}_{(\omega,v)}] = \begin{bmatrix} \widehat{\omega} & \mathbf{0}_3 \\ \widehat{v} & \widehat{\omega} \end{bmatrix}. \tag{15}$$

For efficiency, we recommend to ignore the quadratic terms in these latter matrices, thus resulting in the map

$$\mathrm{C}_{\mathrm{TLN}}((\omega, v)) = \mathbf{I}_6 - \frac{1}{2}[\mathrm{ad}_{(\omega,v)}],$$

which corresponds to the trapezoidal Lie-Newmark scheme [Bou-Rabee and Marsden 2009]. This map can be substituted for either $\mathrm{C}_{\exp}$ or $\mathrm{C}_{\mathrm{cay}}$ *without losing the second-order accuracy* of the discrete Euler-Poincaré equations. The integrator update step is summarized in Fig. 6 and described in pseudocode in Appendix A.

Given $(R_k, x_k, \omega_{k-1}, v_{k-1})$
- pick $\tau = \exp$ or $\tau = \mathrm{cay}$
- compute matrices B_τ and C_τ using (14)-(15)
- Solve (11) for (ω_k, v_k) using a Newton-type solver
- Update (R_{k+1}, x_{k+1}) using (10)

Fig. 6: *Pseudocode of our holonomic integrator for an implicit step.*

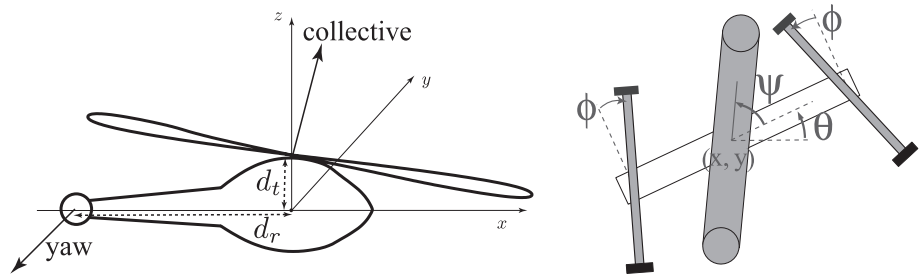


Fig. 7: *Helicopter and Snakeboard: (left) the helicopter model used in our tests; (right) pose & shape space variables of the snakeboard.*

4.3 Examples of Holonomic Vehicles

4.3.1 *Simple helicopter.* Consider the following model of a helicopter depicted in Fig. 7. The vehicle is modeled as a single underactuated rigid body on $SE(3)$ with mass m and principal moments of rotational inertia J_1, J_2, J_3. The inertia tensors are $\mathbb{J} = \mathrm{diag}(J_1, J_2, J_3)$ and $\mathbb{M} = m\mathbf{I}_3$. The system is subject to external force due to gravity

$$f_{\text{ext}}((R, x), (\omega, v)) = (0, 0, 0, R^T(0, 0, -9.81m)).$$

The vehicle is controlled through a *collective* u_c (lift produced by the main rotor) and a *yaw* u_ψ (force produced by the rear rotor), while the direction of the lift is controlled by tilting the main blades forward or backward through a *pitch* γ_p and sideways through a *roll* γ_r. The shape variables are thus $\gamma = (\gamma_p, \gamma_r)$, and the controls are $u = (u_c, u_\psi)$. The resulting control matrix becomes:

$$F(\gamma) = \begin{bmatrix} d_t \sin\gamma_r & 0 \\ d_t \sin\gamma_p \cos\gamma_r & 0 \\ 0 & d_r \\ \sin\gamma_p \cos\gamma_r & 0 \\ -\sin\gamma_r & -1 \\ \cos\gamma_p \cos\gamma_r & 0 \end{bmatrix}$$

The discrete equations of motion are then obtained by substituting these specific expressions into the general integrator derived in Sec. 4.2. Examples of motion generated by this integrator can be found in Figs. 9 and 12.

4.3.2 *Boat subject to external disturbances.* Consider a simple boat modeled as a rigid body subject to gravity, buoyancy, wind forces, and hydrodynamic forces from solid-fluid interaction that typically include drag, waves, and currents. The boat has inertia tensors $\mathbb{J}$ and $\mathbb{M}$, and is controlled by two fixed thrusters placed at the rear of the boat attached at positions $c_r \in \mathbb{R}^3$ and $c_l \in \mathbb{R}^3$ with respect to the body-fixed frame, producing propelling forces u_r and u_l, respectively. The control input vector is $u = (u_r, u_l)$ and there are no shape variables ($M = \emptyset$), resulting in the control matrix

$$F(\gamma) = \begin{bmatrix} 0 & -c_r^3 & -c_r^2 & 1 & 0 & 0 \\ 0 & -c_l^3 & c_l^2 & 1 & 0 & 0 \end{bmatrix}^T$$

External forces acting on the boat of mass m can be modeled as

$$f_{\text{ext}} = 9.81 \begin{bmatrix} R^T((x_s - x) \times (0, 0, \Pi_s)) \\ R^T(0, 0, -m + \Pi_s) \end{bmatrix} + \frac{1}{A} \sum_i \begin{bmatrix} r_i \times f_h((R, x), (\omega, v), r_i) \\ f_h((R, x), (\omega, v), r_i) \end{bmatrix} dA_i, \tag{16}$$

where the scalar Π_s is the volume of the submerged part of the boat (measured in liters) and $x_s \in \mathbb{R}^3$ are the global coordinates of the volume centroid, $r_i \in \mathbb{R}^3$ are the body-fixed coordinates of a point centered at a surface patch with area dA_i of the boundary of the submerged part of the boat, and $f_h : (SO(3) \times \mathbb{R}^3) \times \mathbb{R}^6 \times \mathbb{R}^3 \to \mathbb{R}^3$ is the hydrodynamic

force that depends on the state of the vehicle and the relative position of the patch. The summation provides a boundary-element approximation of the continuous resulting force from the water to the boat, where $A = \sum_i dA_i$ is the total submerged surface area. A simple approximation of the hydrodynamic force f_h is:

$$f_h((R,x),(\omega,v),r_i) = -E(\dot{r}_i - R^T\eta(x+Rr_i)) - Pn_i,$$

where $\dot{r}_i = v + \omega \times r_i$ is the velocity of the i^{th} surface element, $\eta : \mathbb{R}^3 \to \mathbb{R}^3$ is the water fluid velocity expressed in the global frame, E is a positive definite damping matrix, $n_i \in \mathbb{R}^3$ is the outward-pointing unit normal to the vessel surface at point r_i (in body-fixed coordinates), and the scalar P is a fixed magnitude of the normal force (typically due to pressure) exerted by the water. Fig. 8 shows an example of such a boat model coupled with a liquid animation generated using the FLIP method [Zhu and Bridson 2005].

5. NONHOLONOMIC VEHICLES

We now consider a more general class of vehicles whose dynamics are affected by changes in shape (such as cars) and are subject to nonholonomic constraints. Such systems were introduced in Sec. 3.2. In this section we provide discrete equations of motion that can either be used to directly simulate vehicle trajectories, or can be applied as constraints in an optimal control problem. For simplicity we discuss only the rigid motion groups $SE(2)$ and $SE(3)$, though our algorithms are valid for any group G.

5.1 Nonholonomic Integrator

Discrete nonholonomic equations of motion are derived by applying the variational principle expressed in Eq. (6). This principle is defined in terms of the variables $\xi \in \mathfrak{g}$ and $\mu \in \mathfrak{g}^*$ which are elements of the Lie algebra and its dual, respectively. These elements can be represented simply as a vector in $\mathbb{R}^n$ and written in coordinates as $\xi = \xi^i E_i$ and $\mu = \mu_i E^i$ where $\{E_i\}$ is any orthonormal basis for the Lie algebra and $\{E^i\}$ is the corresponding dual basis. As a concrete example, an element $\xi \in \mathfrak{se}(3)$ is encoded as a vector $(\omega, v) \in \mathbb{R}^6$, where ω and v are the angular and linear velocities, respectively (see Eq. (7)).

With this setup, the resulting nonholonomic integrator resulting from the discrete variational principle in Eq. (6) is:

$$\begin{aligned}
&g_{k+1} = g_k\, \tau(h(\Omega_k - \mathcal{A}(r_{k+\alpha}) \cdot u_k)),\\
&r_{k+1} = r_k + hu_k,\\
&[e_1(r_k), ..., e_b(r_k)]^T \,\mathrm{DEP}_\tau(k) = \mathbf{0},\\
&\frac{\partial \ell_{k+\alpha}}{\partial u} - \frac{\partial \ell_{k-1+\alpha}}{\partial u} - h\left(\alpha \frac{\partial \ell_{k-1+\alpha}}{\partial r} + (1-\alpha)\frac{\partial \ell_{k+\alpha}}{\partial r}\right)\\
&= [\mathcal{A}(r_k)]^T \,\mathrm{DEP}_\tau(k) + h\left(\alpha f_{k-1+\alpha} + (1-\alpha) f_{k+\alpha}\right),
\end{aligned} \tag{17}$$

for $b = 1, ..., \dim(\mathfrak{s}_r)$, and $k = 1, ..., N-1$. We have used the notation $\ell_{k+\alpha} := \ell(r_{k+\alpha}, u_k, \xi_k)$ and the *discrete Euler-Poincaré* operator (DEP):

$$\mathrm{DEP}_\tau(k) := \mathrm{C}_\tau^T(h\xi_k)\mu_k - \mathrm{C}_\tau^T(-h\xi_{k-1})\mu_{k-1} - hf_{\text{ext}}(g_k, \xi_{k-1}), \tag{18}$$

where $\xi_k = \Omega_k - \mathcal{A}(r_{k+\alpha}) \cdot u_k$ and $\mu_k = \frac{\partial \ell_{k+\alpha}}{\partial \xi}$. The basis vectors $e_b(r)$ are chosen so that $\mathrm{span}\{e_b(r)\} = \mathfrak{s}_r$. Therefore, the matrix $[e_1(r_k), ..., e_b(r_k)]$ acts as a projection onto the subspace spanned by all body-fixed velocity directions that leave the Lagrangian invariant and that satisfy the constraints. The definition of the map C_τ depends on the group, and the most common case $G = SE(3)$ (which we stick to in this paper) was already explicitly given in Eq. (15) for $\tau = \exp$ and $\tau = \mathrm{cay}$. The function $f_{\text{ext}} : G \times \mathbb{R}^n \to \mathbb{R}^n$ defines any external forces acting on the body such as gravity or friction while forces f_k refer to controls applied to shape variables.

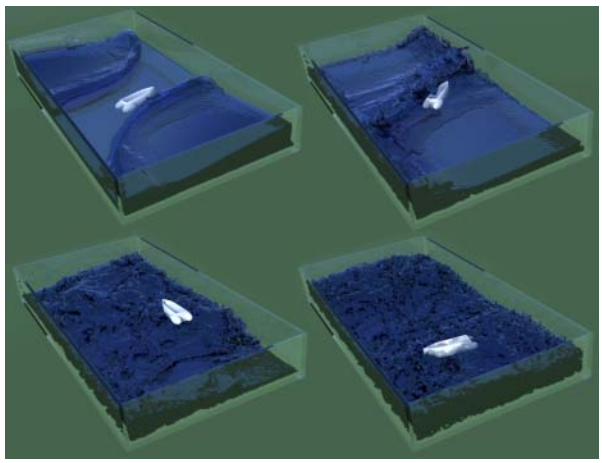


Fig. 8: *Our vehicle integrator can easily be coupled with standard simulation methods for other physical phenomena. Here a holonomic boat with rear thrusters interacts with a turbulent free-surface flow.*

Fig. 9: *Helicopter: in this animation generated with our integrator for holonomic systems, a helicopter is manipulated through a joystick.*

5.2 Examples with Nonholonomic Constraints

5.2.1 *Car with simple dynamics.* We study the kinematic car model defined in [Kelly and Murray 1995] with added simple dynamics. The configuration space is $Q = S^1 \times S^1 \times SE(2)$ with coordinates $q = (\psi, \sigma, \theta, x, y)$, where (θ, x, y) are the orientation and position of the car, ψ is the rolling angle of the rear wheels, and σ is defined by $\sigma = \tan(\phi)$ where ϕ is the steering angle. The car has mass m, rear wheel inertia I, rotational inertia K, and we assume that the steering inertia is negligible. The car is controlled by rear wheels torque f^ψ and steering velocity u^σ. The Lagrangian is then expressed as:

$$\mathcal{L}(q, \dot{q}) = \frac{1}{2}\left(I\dot{\psi}^2 + K\dot{\theta}^2 + m(\dot{x}^2 + \dot{y}^2)\right),$$

and the constraints (see [Kelly and Murray 1995]) are

$$\cos\theta dx + \sin\theta dy = \rho d\psi, \qquad -\sin\theta dx + \cos\theta dy = 0, \qquad d\theta = \frac{\rho}{l}\sigma d\psi,$$

where l is the distance between front and rear wheel axles, and ρ is the radius of the wheels. These constraints simply encode the fact that the car must turn in the direction in which the front wheels are pointing, that the car cannot slide sideways, and that the change in orientation is proportional to the steering angle and turning rate of the wheels, respectively. Note now that the Lagrangian and constraints are invariant to rotation and translation of the orientation and position of the car. The *reduced Lagrangian* can be expressed as

$$\ell(r, u, \xi) = \frac{1}{2}\left(u^T \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} u + \xi^T \begin{bmatrix} K & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix} \xi\right),$$

where we defined $u = (u^\psi, u^\sigma)$, $\xi = -\mathcal{A}(r) \cdot u$, and where the shape coordinates are denoted by $r = (\psi, \sigma)$. For this car model, the matrix representation of the connection $\mathcal{A}$ dependent on r is:

$$[\mathcal{A}(r)] = \begin{bmatrix} -\frac{\rho}{l}\sigma & 0 \\ -\rho & 0 \\ 0 & 0 \end{bmatrix} \tag{19}$$

Car Integrator. The discrete equations of motion are derived by substituting the Lagrangian and the connection of the steered car into (6). If we pick $\tau = \exp$, the equations of motion in Eq. (17) simplify to:

$$\begin{cases} g_{k+1} = g_k \ \exp(-h\mathcal{A}(r_{k+\alpha}) \cdot u_k), \\ r_{k+1} = r_k + hu_k, \\ \dfrac{\partial \ell_{k+\alpha}}{\partial u} - \dfrac{\partial \ell_{k-1+\alpha}}{\partial u} = A(r_k)^T(\mu_k - \mu_{k-1}) + h\left(\alpha f_{k-1+\alpha} + (1-\alpha) f_{k+\alpha}\right), \end{cases}$$

for $k = 1, ..., N-1$. The exponential map exp for $G = SE(2)$ are given in Appendix B. After substituting the Lagrangian and the body-fixed velocity ξ, one can explicitly write the update equations of the configuration variables as follows:

$$x_{k+1} - x_k = \begin{cases} \frac{v_k}{\omega_k}(\sin(\theta_k + h\omega_k) - \sin\theta_k) & \text{if } \omega \neq 0; \\ \cos\theta_k h v_k & \text{if } \omega = 0. \end{cases}$$

$$y_{k+1} - y_k = \begin{cases} \frac{v_k}{\omega_k}(-\cos(\theta_k + h\omega_k) + \cos\theta_k) & \text{if } \omega \neq 0; \\ \sin\theta_k h v_k & \text{if } \omega = 0. \end{cases}$$

$$\theta_{k+1} = \theta_k + h\omega_k,$$

$$\sigma_{k+1} = \sigma_k + hu_k^\sigma,$$

$$(I + \rho^2 m)(u_k^\psi - u_{k-1}^\psi) + \frac{\rho^2 K}{l^2}\sigma_k(\sigma_{k+\alpha}u_k^\psi - \sigma_{k-1+\alpha}u_{k-1}^\psi) = h\left(\alpha f_{k-1+\alpha}^\psi + (1-\alpha) f_{k+\alpha}^\psi\right),$$

where we defined $v_k = \rho u_k^\psi$, and $\omega_k = (\rho/l)\sigma_{k+\alpha}u_k^\psi$. The integrator is easily implemented as it is *fully explicit* for any quadrature choice (i.e., for any $\alpha \in [0,1]$).

5.2.2 *The snakeboard.* The snakeboard is a wheeled board closely resembling the popular skateboard with the main difference that both the front and the rear wheels can be steered *independently*. This feature causes an interesting interplay between momentum conservation and nonholonomic constraints: the rider is able to build up velocity *without pushing off the ground* by transferring the momentum generated by a twist of the torso into motion of the board coupled with steering of the wheels through pivoting of the feet. When the steering wheels stop turning, the systems moves along a circular arc and the momentum around the center of this rotation is conserved. A robotic version of the snakeboard also exists, equipped with a momentum-generating rotor and steering servos [Ostrowski 1996].

The shape space variables of the snakeboard are $r = (\psi, \phi) \in S^1 \times S^1$ denoting the rotor angle and the steering wheels angle, while its configuration is defined by $(\theta, x, y) \in SE(2)$ denoting orientation and position of the board (see Fig. 7(right)), leading to a configuration space $Q = S^1 \times S^1 \times SE(2)$. Additional parameters are its mass m, the distance l from its center to the wheels, and the moments of inertia I and J of the board and the steering. The kinematic constraints of the snakeboard are:

$$\begin{aligned} -l\cos\phi d\theta - \sin(\theta + \phi)dx + \cos(\theta + \phi)dy = 0, \\ l\cos\phi d\theta - \sin(\theta - \phi)dx + \cos(\theta - \phi)dy = 0, \end{aligned}$$

reflecting the fact that the system must move in the direction in which the wheels are pointing and spinning. The constraint distribution is spanned by three covectors:

$$\mathcal{D}_q = \text{span}\left\{\frac{\partial}{\partial\psi}, \frac{\partial}{\partial\phi}, c\frac{\partial}{\partial\theta} + a\frac{\partial}{\partial x} + b\frac{\partial}{\partial y}\right\},$$

where $a = -2l\cos\theta\cos^2\phi,\ b = -2l\sin\theta\cos^2\phi,\ c = \sin 2\phi$. The group directions defining the vertical space are:

$$\mathcal{V}_q = \text{span}\left\{\frac{\partial}{\partial\theta}, \frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right\};$$

therefore, the constrained symmetry space is expressed as:

$$\mathcal{S}_q = \mathcal{V}_q \cap \mathcal{D}_q = \text{span}\left\{ c\frac{\partial}{\partial\theta} + a\frac{\partial}{\partial x} + b\frac{\partial}{\partial y} \right\}. \tag{20}$$

Since $\mathcal{D}_q = \mathcal{S}_q \oplus \mathcal{H}_q$, we have $\mathcal{H}_q = \text{span}\left\{ \frac{\partial}{\partial\psi}, \frac{\partial}{\partial\phi} \right\}$. Finally, the Lagrangian of the system is $\mathcal{L}(q,\dot{q}) = \frac{1}{2}\dot{q}^T\mathbb{M}\,\dot{q}$ where:

$$\mathbb{M} = \begin{bmatrix} I & 0 & I & 0 & 0 \\ 0 & 2J & 0 & 0 & 0 \\ I & 0 & ml^2 & 0 & 0 \\ 0 & 0 & 0 & m & 0 \\ 0 & 0 & 0 & 0 & m \end{bmatrix}.$$

Consequently, the reduced Lagrangian is expressed as: $\ell(r,u,\xi) = (u,\xi)^T\mathbb{M}\,(u,\xi)$. There is only one direction along which snakeboard motions lead to momentum conservation: it is defined by the basis vector:

$$e_1(r) = 2l\cos^2\phi \begin{bmatrix} \frac{\tan\phi}{l} \\ -1 \\ 0 \end{bmatrix},$$

and, thus, there is only one momentum variable $\mu_1 = \left\langle \frac{\partial\ell}{\partial\xi}, e_1(r) \right\rangle$. Using this variable we can derive the connection according to [Ostrowski 1996; Bloch et al. 1996] as:

$$[\mathcal{A}(r)] = \begin{bmatrix} \frac{I}{ml^2}\sin^2\phi & 0 \\ -\frac{I}{2ml}\sin 2\phi & 0 \\ 0 & 0 \end{bmatrix}, \quad \text{and} \quad \Omega = \frac{1}{4ml^2\cos^2\phi}\mu.$$

Snakeboard Integrator. The discrete equations of motion are derived by substituting the particular expression of the Lagrangian and the connection for the snakeboard directly into (6). Selecting $\tau = \exp$ simplifies the equations to:

$$\begin{cases} g_{k+1} = g_k \exp(h(\Omega_k - \mathcal{A}(r_{k+\alpha}) \cdot u_k)), \\ r_{k+1} = r_k + hu_k, \\ e_1(r_k)^T(\mu_k - \mu_{k-1}) = 0, \\ \dfrac{\partial\ell_{k+\alpha}}{\partial u} - \dfrac{\partial\ell_{k-1+\alpha}}{\partial u} = h\left(\alpha f_{k-1+\alpha} + (1-\alpha) f_{k+\alpha}\right), \end{cases}$$

for $k = 1, ..., N-1$. Setting $\xi = (\xi^1, \xi^2, 0)$ computed from $\xi_k = \Omega_k - \mathcal{A}(r_{k+\alpha}) \cdot u_k$ (with $u_k = (u_k^\psi, u_k^\phi)$), the final discrete equations of motion are obtained by substituting:

$$\mu = (ml^2\xi^1 + Iu^\phi, m\xi^2, 0), \qquad \frac{\partial\ell}{\partial u} = (I(u^\psi + \xi^1), 2Ju^\phi),$$

leading to an explicit update for the next state. Fig. 1 shows this integrator in action, where the rider first twists his hip to create momentum, then rotates its feet.

6. OPTIMAL CONTROL OF VEHICLES

Control is an important aspect of computer animation. This is particularly true of vehicle animation: when the vehicle dynamics is complex and underactuated, desirable trajectories might be very difficult to generate by directly applying inputs. A common approach to avoid this delicate tweaking of control inputs is to perform interpolation of keyframes created by the designer or through capturing the joystick movements generated by expert pilots. An alternative approach, called *optimal control*, is to let the computer pick an optimal set of inputs to achieve a particular motion

(see [Popović et al. 2000]—or [Grzeszczuk et al. 1998] for an example of neural network based control). We sketch how to use the integrators we developed to devise a method for automatic generation of natural vehicle motions based on numerical optimization of trajectories that satisfy vehicle dynamics and minimize a user-defined cost function. Our integrators offer a particularly robust and efficient framework for optimal control as they are coordinate-free and use a minimum state dimension discretization, thus avoiding issues of chart-switching and singularities.

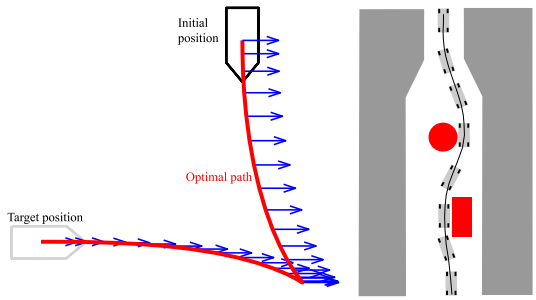


Fig. 10: *Controlling trajectories: (left) optimal docking subject to wind forces (shown as arrows along the path). The boat starts with zero velocity and must arrive at the designated position with zero velocity (docking); (right) a car avoids obstacles on its way, with the least amount of steering.*

Minimize Cost Function
subject to
initial and final poses and velocities,
and: // *for each pose along the path*
for each $k = 1$ to N **do**
- dynamics: eqs. (10)-(11) or eqs. (17)
- pose and velocity bounds
- shape variable bounds, e.g. joint limits
- additional constraints on k-th state

Fig. 11: *Pseudocode of the nonlinear optimization performed for vehicle control: a cost function is minimized under the constraint that each pose along the resulting path satisfies the update equations of our integrator.*

Formulation and Implementation. In this section, we provide a *direct optimal control formulation* that results in a constrained optimization problem of minimizing a chosen cost function subject to the dynamics of a vehicle. Depending on the scenario that the user wishes to achieve, typical cost functions include minimum control effort, minimum time, minimum distance, or safest travel (achieved by augmenting the cost function with a repulsive potential from hazards along the path). Additional constraints such as joint limits and obstacle avoidance can be imposed in the form of inequalities or as penalizing terms in the cost function.

We follow the *Discrete Mechanics and Optimal Control* (DMOC) approach introduced in [Junge et al. 2005], where the dynamics is implemented using the integrators introduced above, i.e., the trajectory of a vehicle is optimized *subject to* a discrete variational principle such as (1) and (6). The overall optimization procedure is described in Fig. 11; we refer the reader to [Junge et al. 2005] for further details regarding the optimization setup. The resulting formulation can be solved using a standard constrained optimization technique such as sequential quadratic programming (SQP). Examples in this section were implemented using the SNOPT package [Gill et al. 2002] since it supports sparse constraint Jacobians which are needed for efficiency. We successfully tested optimal control of all the vehicles presented above: we were able to get a helicopter across a digital canyon while minimizing control effort (see Figs. 12, 9), get a boat to make a docking maneuver automatically, and control a car among obstacles (see Fig. 10).

7. CONCLUSION

We presented a new framework for designing Lie group integrators for a wide assortment of holonomic and nonholonomic vehicles, including cars, boats, snakeboards and helicopters, and discussed how one might design and implement integrators for additional vehicles. We demonstrated that Lie group integrators respect structure and are particularly robust for large time steps, while retaining (sometimes even improving) the efficiency of Runge-Kutta methods. We also showed how these integrators can be applied to optimal control, quickly producing trajectories that fit users' constraints and can later be edited to their liking. In the future we plan to develop hierarchical solvers for optimal control in order to accelerate convergence as well as to offer level-of-detail control of the generated motion.

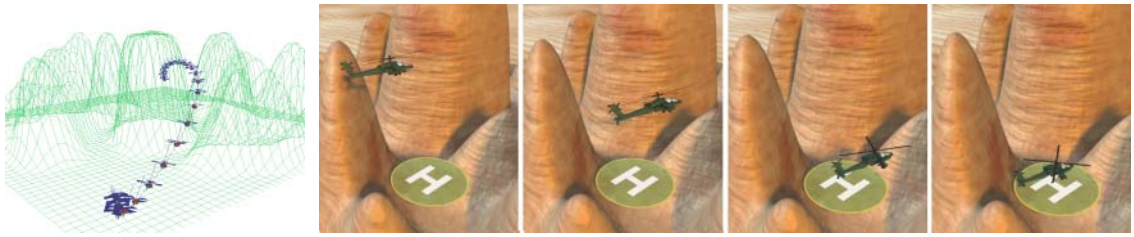


Fig. 12: *Optimized trajectories of a helicopter flying through in a complex canyon (left), and automatically landing on the mark (right).*

APPENDIX

A. PSEUDOCODE

In this first appendix, we provide pseudocode (Matlab-style) of our time integrators for unconstrained systems (**HolInt**) and nonholonomic systems (**NonhInt**). Both integrators are defined for systems for which the vehicle pose is described by $G = SE(3)$. In their general form the integrators are implicit and an update is performed by solving the dynamics equations **Unconstr_Dyn** and **Nonh_Dyn**, respectively. However, these updates can be made explicit for certain systems such as the car and the snakeboard as we presented in 5.2. Moreover, we systematically use the update map $\tau = \text{cay}$ and the operator $\mathrm{C}_{\mathrm{TLN}}$ in this pseudocode. Control forces in the unconstrained case are computed using a matrix function $\mathbf{F}$. External forces are computed using function $\mathbf{f}_{\text{ext}}$. Finally, the nonholonomic integrator code assumes the existence of a matrix function $\mathcal{A}$ (the connection) as well as a symbolic expression for the vector **RDEL** as defined in function **Nonh_Dyn** as we discussed in Sec. 3.2.

Function $[\xi_k, g_{k+1}] = \textbf{HolInt}(\xi_{k-1}, g_k)$
% mass matrix
$\mathbb{I} = \begin{bmatrix} \mathbb{J} & \mathbf{0} \\ \mathbf{0} & \mathbb{M} \end{bmatrix}$
% initial guess (e.g., Euler step)
$\xi_k = \xi_{k-1} + h \cdot \text{inv}(\mathbb{I}) \cdot (\textbf{se3\_ad}\,(h \cdot \xi_k)^t \cdot \mathbb{I} \cdot \xi_{k-1}$
$+\mathbf{f}_{\text{ext}}(\xi_{k-1}, g_k) + \mathbf{F}(\gamma_k)u_k)$
% implicit dynamics solve (e.g. Newton's method)
$\xi_k = \text{fsolve}(\textbf{Unconstr\_Dyn}, \xi_k, \xi_{k-1}, g_k)$
% explicit pose update
$g_{k+1} = g_k \cdot \textbf{se3\_cay}(h\xi_k)$

Function $[u_k, \Omega_k, r_{k+1}, g_{k+1}] = \textbf{NholInt}(u_{k-1}, \Omega_{k-1}, r_k, g_k)$
% initial guess (e.g. Euler step)
$u_k = u_{k-1} + h\,(\alpha f_{k-1+\alpha} + (1-\alpha) f_{k+\alpha})$
$\Omega_k = \Omega_{k-1}$
% implicit dynamics solve (e.g. Newton's method)
$[u_k, \Omega_k] = \text{fsolve}(\textbf{Nonh\_Dyn}, u_k, \Omega_k, u_{k-1}, \Omega_{k-1}, r_k, g_k)$
$\xi_k = \Omega_k - \mathcal{A}(r_k + h \cdot \alpha \cdot u_k) \cdot u_k$
% explicit pose update
$r_{k+1} = r_k + h \cdot u_k$
$g_{k+1} = g_k \cdot \textbf{se3\_cay}(h\xi_k)$

Function $f = \textbf{Nonh\_Dyn}(u_k, \Omega_k, u_{k-1}, \Omega_{k-1}, r_k, g_k)$
$\xi_{k-1} = \Omega_{k-1} - \mathcal{A}(r_k - h \cdot (1-\alpha) \cdot u_{k-1}) \cdot u_{k-1}$
$\xi_k = \Omega_k - \mathcal{A}(r_k + h \cdot \alpha \cdot u_k) \cdot u_k$
$\text{DEP} = \textbf{se3\_DEP}(\xi_k, \xi_{k-1}, g_k)$
$\text{RDEL} = \frac{\partial \ell_{k+\alpha}}{\partial u} - \frac{\partial \ell_{k-1+\alpha}}{\partial u} - h\left(\alpha \frac{\partial \ell_{k-1+\alpha}}{\partial r} + (1-\alpha)\frac{\partial \ell_{k+\alpha}}{\partial r}\right)$
$f = [\,[e_1(r_k), ..., e_b(r_k)]^t \cdot \text{DEP};$
% returns:
$\text{RDEL} - [\mathcal{A}(r_k)]^t \cdot \text{DEP} - h\,(\alpha f_{k-1+\alpha} + (1-\alpha) f_{k+\alpha})\,]$

Function $f = \textbf{Unconstr\_Dyn}(\xi_k, \xi_{k-1}, g_k)$
$f = \textbf{se3\_DEP}(\xi_k, \xi_{k-1}, g_k) - h\mathbf{F}(\gamma_k)u_k$

Function $f = \textbf{se3\_DEP}(\xi_k, \xi_{k-1}, g_k)$
$f = \textbf{se3\_C}_{\textbf{TLN}}(h \cdot \xi_k)^t \cdot \mathbb{I} \cdot \xi_k$
$- \textbf{se3\_C}_{\textbf{TLN}}(-h \cdot \xi_{k-1})^t \cdot \mathbb{I} \cdot \xi_{k-1}$
$-h \cdot \mathbf{f}_{\text{ext}}(g_k, \xi_{k-1})$

Function $f = \textbf{so3\_cay}(\omega)$
$\widehat{\omega} = \textbf{so3\_ad}(\omega)$
$f = \text{eye}(3) + 4/(4 + \text{norm}(\omega)^2) \cdot (\widehat{\omega} + \widehat{\omega} \cdot \widehat{\omega}/2)$

Function $f = \textbf{se3\_cay}(\xi)$
$\omega = \xi(1:3)$
$v = \xi(4:6)$
$B = 2/(4 + \text{norm}(h\omega)^2) \cdot (2 \cdot \text{eye}(3) + \textbf{so3\_ad}(h\omega))$
$f = \begin{bmatrix} \textbf{so3\_cay}(h\omega) & hB \cdot v \\ \mathbf{0} & 1 \end{bmatrix}$

Function $f = \textbf{se3\_C}_{TLN}(\xi)$
$f = \text{eye}(6) - 0.5 \cdot \textbf{se3\_ad}\,(\xi)$

Function $f = \textbf{se3\_ad}(\xi)$
$\omega = \xi(1:3)$
$v = \xi(4:6)$
$f = \begin{bmatrix} \textbf{so3\_ad}(\omega) & \text{zeros}(3) \\ \textbf{so3\_ad}(v) & \textbf{so3\_ad}(\omega) \end{bmatrix}$

Function $f = \textbf{so3\_ad}(\omega)$
$f = \begin{bmatrix} 0 & -\omega(3) & \omega(2) \\ \omega(3) & 0 & -\omega(1) \\ -\omega(2) & \omega(1) & 0 \end{bmatrix}$

B. GROUP DIFFERENCE MAPS ON $SE(2)$

The coordinates of $SE(2)$ are (θ, x, y) with matrix representation $g \in SE(2)$ given by:

$$g = \begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix}. \tag{21}$$

Using the isomorphic map $\widehat{\cdot}: \mathbb{R}^3 \to \mathfrak{se}(2)$ given by:

$$\widehat{v} = \begin{bmatrix} 0 & -v^1 & v^2 \\ v^1 & 0 & v^3 \\ 0 & 0 & 0 \end{bmatrix} \text{ for } v = \begin{pmatrix} v^1 \\ v^2 \\ v^3 \end{pmatrix} \in \mathbb{R}^3,$$

$\{\widehat{e}_1, \widehat{e}_2, \widehat{e}_3\}$ can be used as a basis for $\mathfrak{se}(2)$, where $\{e_1, e_2, e_3\}$ is the standard basis of $\mathbb{R}^3$.
The two maps $\tau : \mathfrak{se}(2) \to SE(2)$ are given by

$$\exp(v) = \begin{cases} \begin{bmatrix} \cos v^1 & -\sin v^1 & \frac{v^2 \sin v^1 - v^3(1-\cos v^1)}{v^1} \\ \sin v^1 & \cos v^1 & \frac{v^2(1-\cos v^1) + v^3 \sin v^1}{v^1} \\ 0 & 0 & 1 \end{bmatrix} & \text{if } v^1 \neq 0 \\ \begin{bmatrix} 1 & 0 & v^2 \\ 0 & 1 & v^3 \\ 0 & 0 & 1 \end{bmatrix} & \text{if } v^1 = 0 \end{cases}$$

$$\mathrm{cay}(v) = \begin{bmatrix} \frac{1}{4+(v^1)^2} \begin{bmatrix} (v^1)^2 - 4 & -4v^1 & -2v^1v^3 + 4v^2 \\ 4v^1 & (v^1)^2 - 4 & 2v^1v^2 + 4v^3 \end{bmatrix} \\ 0 \quad 0 \quad 1 \end{bmatrix}$$

The maps C_τ can be expressed as the 3×3 matrices:

$$\mathrm{C}_{\exp}(v) = \mathbf{I}_3 - \frac{1}{2}[\mathrm{ad}_v] + \frac{1}{12}[\mathrm{ad}_v]^2, \tag{22}$$

$$\mathrm{C}_{\mathrm{cay}}(v) = \mathbf{I}_3 - \frac{1}{2}[\mathrm{ad}_v] + \frac{1}{4}\left[\, v^1 \cdot v \;\; \mathbf{0}_{3\times 2} \,\right], \tag{23}$$

where:

$$[\mathrm{ad}_v] = \begin{bmatrix} 0 & 0 & 0 \\ v^3 & 0 & -v^1 \\ -v^2 & v^1 & 0 \end{bmatrix}.$$

Acknowledgements. The authors wish to thank Jerrold E. Marsden for inspiration and guidance, Gaurav Sukhatme for his assistance and support, and the reviewers for their feedback. This research project was partially funded by the NSF (ITR DMS-0453145, CCF-0811373, and CMMI-0757106), DOE (DE-FG02-04ER25657), the Caltech IST Center for the Mathematics of Information, and Pixar Animation Studios.

REFERENCES

ALEXA, M. 2002. Linear combination of transformations. In *ACM SIGGRAPH*. 380–387.

BLOCH, A. 2003. *Nonholonomic dynamical systems*. Springer.

BLOCH, A. M., KRISHNAPRASAD, P. S., MARSDEN, J. E., AND MURRAY, R. 1996. Nonholonomic mechanical systems with symmetry. *Arch. Rational Mech. Anal.* 136, 21–99.

BOU-RABEE, N. AND MARSDEN, J. E. 2009. Hamilton-Pontryagin integrators on Lie groups Part I: Introduction and structure-preserving properties. To appear in *Foundations of Computational Mathematics*.

BULLO, F. AND LEWIS, A. 2004. *Geometric Control of Mechanical Systems*. Springer.

CELLEDONI, E. AND OWREN, B. 2003. Lie group methods for rigid body dynamics and time integration on manifolds. *Comput. meth. in Appl. Mech. and Eng. 19,* 3,4, 421–438.

CENDRA, H., MARSDEN, J., AND RATIU, T. 2001a. Geometric mechanics, Lagrangian reduction, and nonholonomic systems. In *Mathematics Unlimited-2001 and Beyond*. 221–273.

CENDRA, H., MARSDEN, J. E., AND RATIU, T. S. 2001b. Lagrangian reduction by stages. *Mem. Amer. Math. Soc. 152,* 722, 108.

CORTÉS, J. 2002. *Geometric, Control and Numerical Aspects of Nonholonomic Cystems*. Springer.

CORTÉS, J. AND MARTÍNEZ, S. 2001. Non-holonomic integrators. *Nonlinearity 14,* 5, 1365–1392.

CRAFT ANIMATIONS. Craft Director Tools 4-wheeler (Autodesk Maya plugin).

DE LEON, M., DE DIEGO, D. M., AND SANTAMARIA-MERINO, A. 2004. Geometric integrators and nonholonomic mechanics. *Journal of Mathematical Physics 45,* 3, 1042–1062.

DORMAND, J. 1996. *Numerical Methods for Differential Equations*. CRC Press.

FEDOROV, Y. N. AND ZENKOV, D. V. 2005. Discrete nonholonomic LL systems on Lie groups. *Nonlinearity 18*, 2211–2241.

GILL, P. E., MURRAY, W., AND SAUNDERS, M. A. 2002. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM J. on Optimization 12,* 4, 979–1006.

GRZESZCZUK, R., TERZOPOULOS, D., AND HINTON, G. 1998. Neuroanimator: fast neural network emulation and control of physics-based models. In *Proceedings of ACM SIGGRAPH*. 9–20.

HAIRER, E., LUBICH, C., AND WANNER, G. 2006. *Geometric Numerical Integration*. Number 31 in Springer Series in Computational Mathematics. Springer-Verlag.

ISERLES, A., MUNTHE-KAAS, H. Z., NØRSETT, S. P., AND ZANNA, A. 2000. Lie group methods. *Acta Numerica 9*, 215–365.

JUNGE, O., MARSDEN, J., AND OBER-BLÖBAUM, S. 2005. Discrete mechanics and optimal control. In *Proccedings of IFAC*.

KAUFMAN, D. M., EDMUNDS, T., AND PAI, D. K. 2005. Fast frictional dynamics for rigid bodies. *ACM Trans. Graph. 24,* 3, 946–956.

KELLY, S. AND MURRAY, R. 1995. Geometric phases and robotic locomotion. *Journal of Robotic Systems 12,* 6, 417–431.

KHAREVYCH, L., WEIWEI, TONG, Y., KANSO, E., MARSDEN, J., SCHRÖDER, P., AND DESBRUN, M. 2006. Geometric, variational integrators for computer animation. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*. 1–9.

KINEO CAM. Kineo path planner (standalone automatic path planning software tool). http://www.kineocam.com/.

KOBILAROV, M. 2007. *Discrete Geometric Motion Control of Autonomous Vehicles*. Masters thesis, University of Southern California.

KOON, W.-S. AND MARSDEN, J. E. 1997. Optimal control for holonomic and nonholonomic mechanical systems with symmetry and Lagrangian reduction. *SIAM Journal on Control and Optimization 35,* 3, 901–929.

KRYSL, P. AND ENDRES, L. 2005. Explicit Newmark/Verlet algorithm for time integration of the rotational dynamics of rigid bodies. *International Journal For Numerical Methods In Engineering 62,* 15, 2154–2177.

LANCZOS, C. 1949. *Variational Principles of Mechanics*. University of Toronto Press.

LATOMBE, J.-C. 1991. *Robot Motion Planning*. Kluwer Academic Press.

MARSDEN, J. AND WEST, M. 2001. Discrete mechanics and variational integrators. *Acta Numerica 10*, 357–514.

MARSDEN, J. E. AND RATIU, T. S. 1999. *Introduction to Mechanics and Symmetry*. Springer.

MCLACHLAN, R. AND PERLMUTTER, M. 2006. Integrators for nonholonomic mechanical systems. *Journal of NonLinear Science 16*, 283–328.

MURRAY, R. M., LI, Z., AND SASTRY, S. S. 1994. *A Mathematical Introduction to Robotic Manipulation*. CRC.

OSTROWSKI, J. 1996. The mechanics and control of undulatory robotic locomotion. Ph.D. thesis, California Institute of Technology.

POPOVIĆ, J., SEITZ, S. M., ERDMANN, M., POPOVIĆ, Z., AND WITKIN, A. 2000. Interactive manipulation of rigid body simulations. In *Proceedings of ACM SIGGRAPH*. 209–217.

SIMO, J. C., TARNOW, N., AND WONG, K. K. 1992. Exact energy-momentum conserving algorithms and symplectic schemes for nonlinear dynamics. *Computer Methods in Applied Mechanics and Engineering 100*, 63–116.

STERN, A. AND DESBRUN, M. 2006. Discrete geometric mechanics for variational time integrators. In *ACM SIGGRAPH Course Notes: Discrete Differential Geometry*. 75–80.

ZHU, Y. AND BRIDSON, R. 2005. Animating sand as a fluid. In *Proceedings of ACM SIGGRAPH*. 965–972.