Learning to Imitate via Flow Matching with Physics-Constrained Vector Fields

Abstract

We present a method for imitation learning from demonstrations that may contain dynamically infeasible examples. Our approach combines flow matching with the Affine Geometric Heat Flow (AGHF), a physics-based vector field that drives trajectories toward dynamic feasibility. We learn a cost functional whose gradient supplements the AGHF field. At inference, trajectory generation requires only a single ODE integration.

1 Background: The Affine Geometric Heat Flow

1.1 Trajectory Optimization for Robotic Systems

Consider a robotic system with configuration $q(t) \in \mathbb{R}^N$ governed by the manipulator equation:

$$H(q(t))\ddot{q}(t) + C(q(t), \dot{q}(t)) = Bu(t) \tag{1}$$

where $H(q(t))$ is the mass matrix, $C(q(t), \dot{q}(t))$ captures Coriolis and gravitational terms, B is the actuation matrix, and $u(t) \in \mathbb{R}^m$ is the control input.

Defining the state $x(t) = [q(t)^\top, \dot{q}(t)^\top]^\top \in \mathbb{R}^{2N}$, we write the dynamics in control-affine form:

$$\dot{x}(t) = F_d(x(t)) + F(x(t))u(t) \tag{2}$$

where

$$F_d(x(t)) = \begin{bmatrix} \dot{q}(t) \\ -H^{-1}(q(t))C(q(t), \dot{q}(t)) \end{bmatrix}, \quad F(x(t)) = \begin{bmatrix} 0 \\ H^{-1}(q(t))B \end{bmatrix}. \tag{3}$$

The trajectory optimization problem seeks a trajectory $\gamma : [0, 1] \to \mathbb{R}^{2N}$ (reparameterized to the unit interval) satisfying boundary conditions $\gamma(0) = x_0$ and $\gamma(1) = x_f$, the dynamics, and minimizing some cost functional.

1.2 Notation

We carefully distinguish between:

- A **trajectory** $\gamma : [0, 1] \to \mathbb{R}^{2N}$, which is a function mapping time to states
- A **state** $\gamma(t) \in \mathbb{R}^{2N}$, which is the value of the trajectory at a specific time t
- A **Lagrangian** $L : \mathbb{R}^{2N} \times \mathbb{R}^{2N} \to \mathbb{R}$, which is a function of a state-velocity pair
- An **action functional** $\mathcal{A}$, which takes a trajectory and returns a scalar

For a state $x \in \mathbb{R}^{2N}$, we write $x_{P1} \in \mathbb{R}^N$ and $x_{P2} \in \mathbb{R}^N$ for the position and velocity components, respectively.

1.3 The Action Functional

The Affine Geometric Heat Flow (AGHF) poses trajectory optimization as minimizing an action functional over trajectories.

Definition 1 (Lagrangian)**.** *The* ***Lagrangian*** $L : \mathbb{R}^{2N} \times \mathbb{R}^{2N} \to \mathbb{R}$ *maps a state-velocity pair* (x, v) *to a scalar:*

$$L(x, v) = k_d \left\| v_{P1} - x_{P2} \right\|^2 + c(x, v) \tag{4}$$

where:

- x_{P2} *denotes the velocity component of state* x
- v_{P1} *denotes the position component of velocity* v
- *The term* $\|v_{P1} - x_{P2}\|^2$ *penalizes* ***kinematic inconsistency***
- $c : \mathbb{R}^{2N} \times \mathbb{R}^{2N} \to \mathbb{R}$ *is a cost function encoding task objectives*
- $k_d > 0$ *controls the strength of the feasibility enforcement*

Definition 2 (Action Functional)**.** *The* ***action functional*** $\mathcal{A}$ *maps a trajectory* γ *to a scalar:*

$$\mathcal{A}(\gamma) = \int_0^1 L(\gamma(t), \dot{\gamma}(t))\, \mathrm{d}t \tag{5}$$

Remark 1. *The kinematic consistency term* $\|\dot{\gamma}_{P1}(t) - \gamma_{P2}(t)\|^2$ *vanishes if and only if the velocity states equal the position derivatives at time* t*. For trajectories satisfying this constraint everywhere, the action reduces to* $\int_0^1 c(\gamma(t), \dot{\gamma}(t))\, \mathrm{d}t$*.*

1.4 The AGHF Partial Differential Equation

AGHF finds optimal trajectories by solving a parabolic PDE that evolves an initial guess in a fictitious time s:

$$\frac{\partial \gamma}{\partial s}(t, s) = -G(\gamma(t, s))^{-1} \left[\frac{\partial L}{\partial x}\big(\gamma(t, s), \dot{\gamma}(t, s)\big) - \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L}{\partial v}\big(\gamma(t, s), \dot{\gamma}(t, s)\big) \right] \tag{6}$$

where $G : \mathbb{R}^{2N} \to \mathbb{R}^{2N \times 2N}$ is a positive-definite metric, and $\frac{\partial L}{\partial x}$ and $\frac{\partial L}{\partial v}$ denote the partial derivatives of the Lagrangian with respect to its first and second arguments.

The boundary conditions $\gamma(0, s) = x_0$ and $\gamma(1, s) = x_f$ are held fixed for all s.

Lemma 1 (Action Decrease)**.** *Along solutions of the AGHF PDE:*

$$\frac{\mathrm{d}\mathcal{A}(\gamma(\cdot, s))}{\mathrm{d}s} \leq 0 \tag{7}$$

with equality if and only if $\frac{\partial \gamma}{\partial s} = 0$ *(steady state).*

The AGHF thus defines a gradient flow on trajectory space that:

1. Monotonically decreases the action
2. Drives trajectories toward dynamic feasibility (via the k_d term)
3. Minimizes the cost c subject to feasibility

1.5 Vector Field on Trajectory Space

Let Γ denote the space of trajectories $\gamma : [0, 1] \to \mathbb{R}^{2N}$ satisfying the boundary conditions $\gamma(0) = x_0$ and $\gamma(1) = x_f$.

Definition 3 (Physical Action)**.** *The* ***physical action*** $A_{phys} : \Gamma \to \mathbb{R}$ *is the action functional with only the kinematic consistency term:*

$$A_{phys}(\gamma) = \int_0^1 k_d \left\| \dot{\gamma}_{P1}(t) - \gamma_{P2}(t) \right\|^2 \mathrm{d}t \tag{8}$$

This measures the total kinematic inconsistency of a trajectory.

Definition 4 (AGHF Vector Field)**.** *The* ***AGHF vector field*** $v_{phys} : \Gamma \to T\Gamma$ *is:*

$$v_{phys}(\gamma) = -G(\gamma)^{-1} \nabla_\gamma A_{phys}(\gamma) \tag{9}$$

where $\nabla_\gamma A_{phys}(\gamma)$ *is the functional gradient of the physical action with respect to the trajectory.*

This vector field has a crucial property: *it always points toward reduced kinematic inconsistency.* Following this field from any initial trajectory eventually yields a dynamically feasible trajectory.

2 Problem: Imitation from Partially Feasible Demonstrations

2.1 Problem Setting

We are given:

- A collection of demonstration trajectories $\{\gamma^{(i)}_{\text{demo}}\}_{i=1}^N$
- These are **state trajectories only**—no control inputs are provided
- The robot dynamics are known
- **Crucially**: Some demonstrations are dynamically feasible for the robot; others are not

We seek to:

1. Identify which demonstrations are feasible (without explicit labels)
2. Learn a generative model that produces trajectories resembling the feasible demonstrations
3. Guarantee that generated trajectories are dynamically feasible

2.2 Challenges

Standard imitation learning methods assume all demonstrations are achievable. When trained on a mixture of feasible and infeasible demonstrations:

- Behavior cloning learns to imitate infeasible behaviors
- The learned policy attempts impossible motions at execution time
- There is no mechanism to distinguish good from bad demonstrations

2.3 Key Insight

We propose to use the AGHF vector field as a component of the generative model. By combining the physics-based AGHF field with a learned cost functional, we can:

1. Generate trajectories that are biased toward feasibility
2. Learn preferences among trajectories from demonstrations

3 Approach: Flow Matching with Physics-Constrained Vector Fields

3.1 Generative Model Structure

We model trajectory generation as integrating a vector field on trajectory space. Starting from a noise trajectory $\gamma_1 \sim \pi$ (e.g., a Brownian bridge prior connecting x_0 to x_f), we integrate:

$$\frac{\mathrm{d}\gamma}{\mathrm{d}s} = v_\theta(\gamma), \quad s : 1 \to 0 \tag{10}$$

to produce a trajectory γ_0.

3.2 Decomposition of the Vector Field

The vector field consists of two components:

$$\boxed{v_\theta(\gamma) = v_{\mathrm{phys}}(\gamma) + v_{\mathrm{learn}}(\gamma)} \tag{11}$$

Physical Component (Fixed). The AGHF vector field defined in Section 1.5:

$$v_{\mathrm{phys}}(\gamma) = -G(\gamma)^{-1}\nabla_\gamma A_{\mathrm{phys}}(\gamma) \tag{12}$$

This is determined entirely by the known robot dynamics and is not learned. It always drives the trajectory toward dynamic feasibility.

Learned Component. We parameterize the learned component directly as a vector field on trajectory space.

Definition 5 (Learned Vector Field)**.** *The **learned vector field** $v_{learn} : \Gamma \to T\Gamma$ is parameterized by θ (e.g., a neural network). We write:*

$$v_{learn}(\gamma; \theta) \tag{13}$$

This vector field captures preferences among trajectories that are not determined by physics alone.

Remark 2. *One could require v_{learn} to be a gradient field by parameterizing a scalar functional $J_\theta : \Gamma \to \mathbb{R}$ and setting $v_{learn} = -\nabla_\gamma J_\theta$. However, since we only ever use the vector field (never the scalar), we can parameterize v_{learn} directly without this constraint.*

4 Training via Flow Matching

4.1 Flow Matching Objective

Flow matching trains a vector field by regression along interpolated paths between data and noise.

Training Data Construction. For each training step, sample:

- A demonstration trajectory $\gamma_{\text{demo}} \in \Gamma$
- A noise trajectory $\gamma_{\text{noise}} \sim \pi$
- A flow time $s \sim \text{Uniform}[0, 1]$

Construct the interpolated trajectory $\gamma_s \in \Gamma$ pointwise:

$$\gamma_s(t) = (1 - s)\gamma_{\text{demo}}(t) + s\,\gamma_{\text{noise}}(t), \quad \forall t \in [0, 1] \tag{14}$$

The **target velocity** is the element of $T\Gamma$ given pointwise by:

$$v^*(t) = \gamma_{\text{demo}}(t) - \gamma_{\text{noise}}(t), \quad \forall t \in [0, 1] \tag{15}$$

This is the velocity that would transport γ_{noise} to γ_{demo} along a straight line in Γ.

4.2 Loss Function

For a demonstration trajectory γ_{demo}, noise trajectory γ_{noise}, and flow time s, the loss is:

$$\boxed{\mathcal{L}(\theta; \gamma_{\text{demo}}, \gamma_{\text{noise}}, s) = \|v_\theta(\gamma_s) - v^*\|^2} \tag{16}$$

where the norm is on $T\Gamma$ (e.g., an L^2 norm over time).

Expanding the vector field:

$$\mathcal{L}(\theta; \gamma_{\text{demo}}, \gamma_{\text{noise}}, s) = \|v_{\text{phys}}(\gamma_s) + v_{\text{learn}}(\gamma_s; \theta) - v^*\|^2 \tag{17}$$

We minimize the average of this loss over the demonstration dataset and sampled noise trajectories and flow times.

5 Required Derivatives for Training

Training the learned vector field $v_{\text{learn}}(\cdot; \theta)$ via stochastic gradient descent requires computing $\frac{\partial \mathcal{L}}{\partial \theta}$. For computation, we discretize trajectories: let $\gamma \in \mathbb{R}^d$ represent the waypoints of a discretized trajectory. The functional gradients become finite-dimensional gradients, and integrals become sums via numerical quadrature.

5.1 Evaluating the Learned Vector Field

The learned vector field $v_{\text{learn}}(\gamma; \theta) \in \mathbb{R}^d$ is the output of a neural network that takes a discretized trajectory as input. Standard backpropagation gives $\frac{\partial v_{\text{learn}}}{\partial \theta}$.

5.2 Gradient of the Physical Action

We require:

$$\nabla_\gamma A_{\text{phys}}(\gamma_s) \in \mathbb{R}^d \tag{18}$$

This is **not learned**, but must be differentiable since it appears in the loss (through v_{phys}). Computing it involves:

- Derivatives of the kinematic consistency term with respect to each waypoint
- Derivatives through the numerical differentiation used to approximate $\dot{\gamma}$

These computations are exactly what AGHF-based methods (e.g., BLAZE) are optimized to perform efficiently using spatial vector algebra.

5.3 Summary of Derivative Requirements

Object	Derivative Required	Notes
Learned vector field v_{learn}	$\frac{\partial v_{\text{learn}}}{\partial \theta}$	Standard backprop
Physical action A_{phys}	$\nabla_\gamma A_{\text{phys}}(\gamma)$	Fixed (not learned)

Table 1: Derivatives required for training.

6 Inference

After training, generating a trajectory is straightforward:

Algorithm 1 Trajectory Generation

Require: Boundary conditions x_0, x_f; learned parameters θ

1: Sample noise trajectory $\gamma_1 \sim \pi(x_0, x_f)$

2: Integrate ODE: $\frac{\mathrm{d}\gamma}{\mathrm{d}s} = -v_\theta(\gamma)$ from $s = 1$ to $s = 0$

3: **return** γ_0

Key properties:

- **No optimization:** Generation requires only a single ODE integration
- **Physics-informed:** The AGHF component biases trajectories toward dynamic feasibility
- **Demonstration-like:** The learned cost guides trajectories toward behaviors present in demonstrations

7 Discussion

7.1 What the Network Learns

The learned vector field v_{learn} captures *preferences among trajectories*—the aspects of demonstration behavior that physics alone does not determine. This may include:

- Preferred velocity profiles
- Smoothness characteristics
- Task-specific spatial preferences

7.2 Relationship to Inverse Optimal Control

Our approach can be viewed as a form of inverse optimal control: we learn a cost function such that optimizing it (via AGHF) recovers demonstration-like behavior. The key differences from classical inverse optimal control are:

1. We use flow matching rather than maximum entropy or margin-based objectives
2. Inference does not require solving an optimization problem

7.3 Computational Considerations

The primary computational costs are:

1. **Training:** Evaluating $v_{\text{phys}}(\gamma)$ and $v_{\text{learn}}(\gamma; \theta)$ at each step, plus standard backpropagation for the parameter update

2. **Inference:** Integrating the ODE, which requires evaluating the vector field at each integration step

Efficient computation of $v_{\text{phys}} = -G^{-1}\nabla_\gamma A_{\text{phys}}$ can leverage spatial vector algebra and rigid body dynamics algorithms, as developed in the BLAZE literature.