CEA Comité Español de Automática
XLVII JORNADAS DE AUTOMÁTICA CÓRDOBA 2026
UNIVERSIDADE DA CORUÑA

Jornadas de Automática

Teaching Hidden Markov Models Through Multidimensional Trajectory Learning: A Hands-On MATLAB Journey

Juliana Manrique-Cordoba[a], Marina Poveda-Pérez[a], Miguel Ángel de la Casa Lillo[a], José María Sabater-Navarro[a,*]

[a] *Bioengineering Institute, Miguel Hernández University of Elche, Avda de la Universidad sn, 03202, Elche, Spain.*

To cite this article: Manrique-Cordoba, J., Poveda-Pérez, M., De la Casa Lillo, M.A., Sabater-Navarro, J.M. 2026. Teaching Hidden Markov Models Through Multidimensional Trajectory Learning: A Hands-On MATLAB Journey. Jornadas de Automática, 47. https://doi.org/10.17979/ja-cea.2026.47.13849

Resumen

Los modelos ocultos de Markov (HMM) son una herramienta ampliamente utilizada en robótica, procesamiento del lenguaje, reconocimiento de voz y procesamiento de señales. Sin embargo, su enseñanza en cursos de ingeniería sigue siendo un reto debido a la brecha entre el formalismo matemático y las aplicaciones realistas. Los recursos existentes se basan en derivaciones abstractas o en bibliotecas de software de tipo caja negra que ocultan el proceso de modelado, dejando a los estudiantes sin las herramientas conceptuales necesarias para adaptar los HMM a problemas novedosos. Este artículo presenta un tutorial de libre acceso en MATLAB LiveScript que guía a los estudiantes a través del proceso completo de los HMM para el aprendizaje de trayectorias: desde la generación de datos sintéticos y la adición de ruido, pasando por la simplificación geométrica y la codificación simbólica, hasta el entrenamiento del modelo y la reconstrucción basada en Viterbi. Cada etapa expone una capa conceptual de forma transparente, produciendo un resultado intermedio examinable. El diseño se basa en principios establecidos de la educación en ingeniería, incluyendo la instrucción guiada, la reducción de la carga cognitiva innecesaria y el aprendizaje activo mediante la exploración de parámetros. El tutorial está disponible públicamente y puede implementarse de inmediato en cursos de grado y posgrado en inteligencia artificial, robótica y procesamiento de datos.

Palabras clave: Hidden Markov Models, Educación, Ejemplos de Referencia, Algoritmos de Aprendizaje.

Abstract

Hidden Markov Models (HMMs) are a tool widely used in robotics, language processing, speech recognition and signal processing, yet their instruction in engineering courses remains challenging due to the gap between mathematical formalism and realistic applications. Existing resources either rely on abstract derivations or black-box software libraries that obscure the modeling pipeline, leaving students without the conceptual tools to adapt HMMs to novel problems. This paper presents an publicly available MATLAB LiveScript tutorial that guides students through the complete HMM pipeline for trajectory learning: from synthetic data generation and noise augmentation, through geometric simplification and symbolic encoding, to model training and Viterbi-based reconstruction. Each stage exposes one conceptual layer transparently, producing an inspectable intermediate result. The design is grounded in established engineering education principles, including guided instruction, reduction of unnecessary cognitive load, and active learning through parameter exploration. The tutorial is publicly available and immediately deployable in undergraduate and graduate courses in artificial intelligence, robotics, and data processing.

Keywords: Hidden Markov Models, Education, Benchmark Examples, Learning Algorithms.

*Correspondence author: j.sabater@umh.es

1. Introduction

Hidden Markov Models (HMMs) are powerful statistical tools for modeling sequential data, with well-established applications in signal processing, speech recognition, and artificial intelligence (Blunsom, 2004). In robotics, HMMs have proven effective for modeling the temporal structure of motion trajectories, supporting tasks such as gesture recognition, skill learning from demonstration, and motion segmentation (Blunsom, 2004). Despite their relevance, teaching HMMs in undergraduate and graduate engineering courses remains a persistent challenge, particularly when the goal extends beyond discrete toy examples to realistic applications such as Learning from Demonstration (LfD) in robotics. Students may associate artificial intelligence with abstract mathematical formulations, which hinders their understanding of how these models operate in practice (Rossiter et al., 2020). Furthermore, the numerical difficulties inherent to HMM inference, such as floating-point underflow in long observation sequences, demand mastery of advanced techniques such as log-probability scaling, adding a significant implementation burden that can obscure the conceptual purpose of the model (Blunsom, 2004; MathWorks, 2024b).

A deeper gap lies in the availability of educational resources that bridge HMM theory with realistic engineering applications. While theoretical foundations are well covered in the literature, instructors lack structured, executable tools that guide students through the full modeling pipeline, from raw, multi-dimensional data to a trained and interpretable model, within a single, coherent learning experience. This gap is particularly pronounced in robotics, where the complexity of real sensor data demands a progression from controlled examples to generalizable workflows (Bellas et al., 2024; Rossiter et al., 2020).

This paper addresses this gap by presenting a structured, publicly available MATLAB tutorial designed to support HMM instruction in engineering curricula. The tutorial is built upon an implementation previously developed by the authors for trajectory learning in robotics (Manrique-Cordoba et al., 2025), which introduced an N-dimensional Douglas–Peucker based simplification algorithm for symbolic trajectory encoding. The present work does not introduce new algorithmic contributions; rather, it presents a pedagogical framework constructed around that publicly available implementation, with the explicit goal of facilitating its adoption in university courses. The choice of MATLAB as the deployment platform is deliberate: widely adopted in engineering curricula (Galli et al., 2017), its LiveScript format enables the creation of interactive, narrative documents that tightly integrate executable code, visual outputs, and formatted explanations in a single pedagogical artifact (MathWorks, 2024a).

2. Background

2.1. Teaching HMMs in Engineering Education

The integration of probabilistic sequence models into engineering education has been identified as both necessary and difficult. Rossiter et al. (2020), in a broad international survey on introductory systems and control courses, found that instructors consistently struggle to shift student focus from procedural computation toward model-based reasoning and uncertainty management. Extending this logic to the field of probabilistic sequence models, this tension is particularly acute for HMMs, where the interplay among the transition matrix, the emission matrix, and the initial state distribution can appear opaque without a compelling, executable example that makes the probabilistic structure tangible.

From the robotics education perspective, Bellas et al. (2024) highlight that topics such as perception and machine learning receive limited formal treatment in curricula, in part because of the difficulty of handling real-time data and environmental uncertainty in controlled classroom settings. This motivates the use of synthetic, controllable datasets as a pedagogical stepping stone: they reproduce the statistical structure of real sensor data while granting the instructor full authority over noise levels, trajectory geometry, and dimensionality.

2.2. Limitations of Existing Approaches

Three categories of existing resources are commonly employed to teach HMMs, each with recognized pedagogical limitations.

Theoretical lecture notes and textbooks. Classic references such as Blunsom (2004) provide rigorous derivations of the forward–backward algorithm, the Viterbi decoder, and the Baum–Welch training procedure. While mathematically complete, these resources rarely provide executable, multi-dimensional examples, leaving a substantial gap between formulation and application.

Black-box software libraries. Tools such as `hmmlearn` (Python) or the MATLAB Statistics and Machine Learning Toolbox (MathWorks, 2024b) lower the implementation barrier but abstract away the pipeline internals. Students who rely solely on these interfaces often develop a superficial understanding of the model, unable to diagnose failures or adapt the approach to novel data modalities.

Single-domain tutorials. Many publicly available HMM tutorials are tailored to speech or natural language processing, with discrete, one-dimensional observation sequences. This framing does not transfer naturally to robotics or motion analysis, where observations are continuous, multi-dimensional, and geometrically structured.

2.3. The Proposed Tutorial as a Pedagogical Bridge

The tutorial presented in this paper addresses the aforementioned limitations through three design decisions. First, it uses synthetically generated 2D and 3D square trajectories as a controlled yet geometrically meaningful benchmark, making the transition from continuous sensor data to symbolic HMM observations explicit and inspectable at each stage. Second, every processing step is exposed as a transparent, executable block within a MATLAB LiveScript document, rather than hidden behind library calls (see Figure 1). Finally, the narrative structure of the LiveScript promotes active learning (Rossiter et al., 2020): students are invited to modify noise amplitudes, simplification thresholds, and codebook resolution, directly observing how these choices propagate through the pipeline and affect the final reconstruction quality.

The algorithmic core of the tutorial, specifically the N–dimensional Douglas–Peucker based simplification and the symbolic encoding scheme, was formally introduced

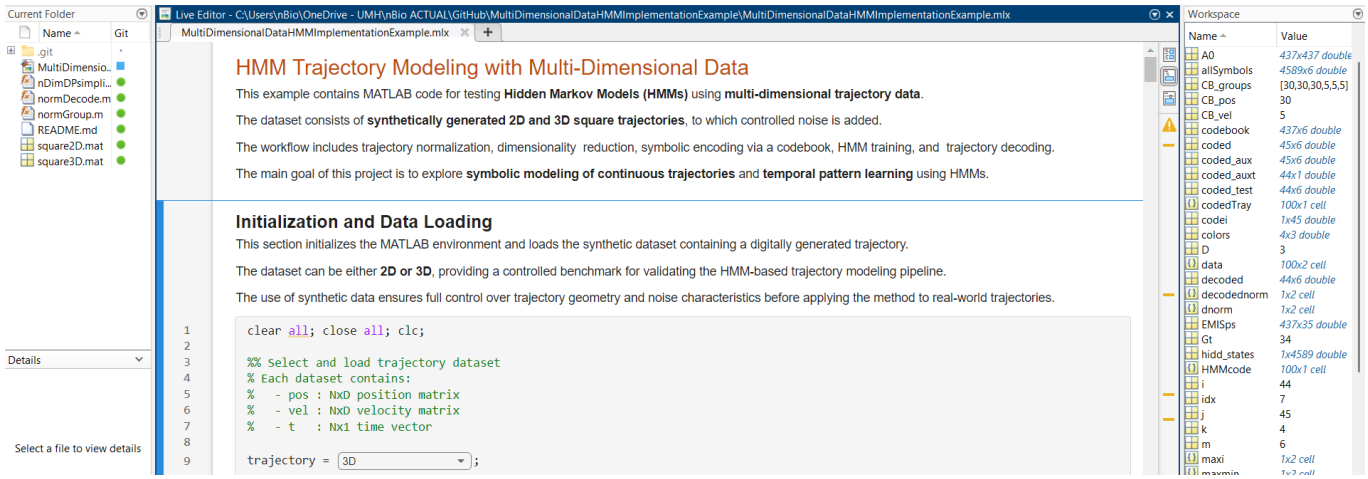

Figure 1: MATLAB LiveScript tutorial.

by Manrique-Cordoba et al. (2025) as a technical contribution to robotic trajectory learning. The present work repurposed that implementation as the backbone of a pedagogical framework, contributing the instructional design, the layered progression of concepts, and the contextualization within engineering education practice.

3. Methods: Tutorial Design

The tutorial MultiDimensional Data HMM Implementation Example is publicly available on MATLAB Central File Exchange (Manrique-Cordoba, 2026). It requires MATLAB with the Statistics and Machine Learning Toolbox and can be downloaded and executed directly within the MATLAB Live Editor environment.

3.1. A Layered Learning Pipeline

The tutorial is structured as a six-stage pipeline in which each stage introduces a single conceptual layer before the next one is revealed. Students encounter complexity incrementally, with each block building directly on the outputs and concepts of the previous one. The complete pipeline is implemented as a MATLAB LiveScript, where executable code, visual outputs, and explanatory narrative coexist in a single scrollable document. Figure 2 illustrates the overall structure.

The six stages are: (1) controlled synthetic data generation, (2) noise augmentation, (3) group-wise normalization and geometric simplification, (4) codebook construction and symbolic encoding, (5) HMM training, and (6) Viterbi decoding and trajectory reconstruction. Each stage is self-contained and can be paused, inspected, or modified independently, supporting both guided laboratory sessions and open-ended exploration.

3.2. Controlled Synthetic Data

The tutorial begins with synthetically generated square trajectories in either two or three spatial dimensions (`square2D.mat`, `square3D.mat`). This design choice is deliberate and pedagogically motivated. Unlike real sensor recordings, synthetic trajectories give the instructor complete control over geometry, scale, and noise characteristics, eliminating confounding factors that would otherwise obscure the relationship between data and model's behavior. Students can immediately verify whether the pipeline produces geometrically coherent outputs, since the ground truth is fully known.

The use of a closed square as a simple, recognizable shape also serves a communicative purpose: it makes the symbolic encoding and reconstruction steps intuitively interpretable, because deviations from the expected geometry are immediately visible in the output plots. This connects the abstract machinery of the HMM to a tangible, inspectable result from the very first stage.

3.3. Noise Augmentation

A single noiseless reference trajectory has limited value for training a probabilistic model. To address this, the tutorial generates multiple independent noisy realizations of the reference trajectory, each treated as a distinct training demonstration. This procedure emulates the variability inherent in real sensing systems, such as motion capture or robotic encoders, without requiring access to physical hardware.

Noise amplitudes are parameterized as fixed percentages of the dynamic range of each signal: `np` controls positional noise (default 5%) and `nv` controls velocity noise (default 1%). This scale-relative formulation ensures that the injected perturbations remain proportional to the signal magnitude, making the parameters interpretable and transferable across trajectories with different units or spatial extents.

3.4. Group-wise Normalization and Geometric Simplification

Before symbolic encoding can proceed, the continuous trajectories must be normalized and compacted into a representation that retains geometric structure while discarding redundant points. This stage applies two sequential operations.

First, a group-wise normalization scheme (`normGroup.m`) scales each feature dimension according to user-defined weights `w`, ensuring that heterogeneous signals, such as Cartesian positions and velocities, contribute proportionally to subsequent processing steps. This prevents high-magnitude dimensions from dominating the geometric simplification and symbolic encoding.

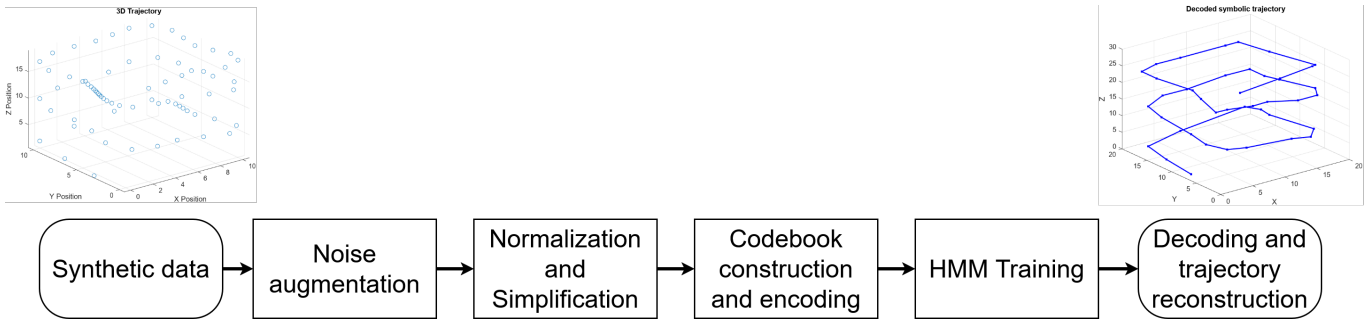

Figure 2: The six-stage layered learning pipeline implemented in the MATLAB LiveScript tutorial. Each block introduces one conceptual layer and produces an inspectable intermediate output that students can examine before proceeding to the next stage.

Second, the N–dimensional Douglas–Peucker based algorithm (`nDimDPsimplification.m`) is applied to each normalized trajectory (Manrique-Cordoba et al., 2025). The algorithm recursively removes points whose orthogonal deviation from the piecewise linear approximation falls below a user-defined `threshold`, retaining only the geometrically representative keypoints. The procedure is dimension-agnostic, operating identically on 2D and 3D trajectories. The resulting simplified trajectories are substantially less densely sampled than the originals, dramatically reducing the size of the symbolic alphabet and the computational cost of HMM training.

This stage is particularly valuable pedagogically because it makes the continuous-to-discrete transition explicit and inspectable. Students can visualize the original normalized trajectory alongside its simplified keypoint representation, directly observing how the choice of `threshold` governs the trade-off between geometric fidelity and representational compactness. Figure 3 illustrates this effect.

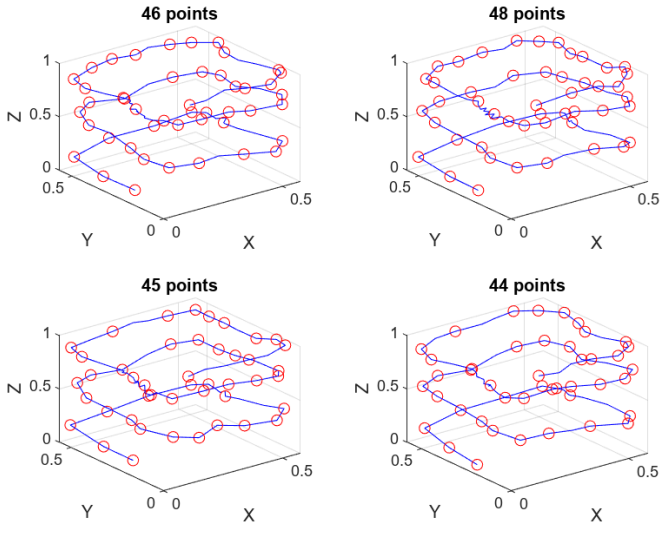

Figure 3: Effect of the N–dimensional Douglas-Peucker based simplification on a 3D square trajectory (71 points). Each subplot shows one simplified trajectory, where the original dense sampling is reduced to a sparse set of keypoints (markers) governed by the `threshold` parameter.

3.5. Codebook Construction and Symbolic Encoding

Standard discrete HMMs require observations drawn from a finite symbolic alphabet. This stage bridges the gap between the continuous geometric representation and the discrete model, and it constitutes the most critical conceptual step in the pipeline.

Each dimension of the simplified trajectories is uniformly discretized according to the codebook resolution parameter `CB_groups`, mapping continuous-valued feature vectors to multi–dimensional integer indices. A data-driven codebook is then constructed by retaining exclusively those discrete symbols that appear in the training data. This approach avoids allocating probability mass to geometrically unreachable regions of the feature space, yielding a compact alphabet whose size is determined by the data rather than by a predetermined grid.

Each simplified trajectory is subsequently encoded as a sequence of codebook indices, producing the symbolic observation sequences that will serve as the hidden states of the HMM. The temporal progression of each trajectory is independently discretized and used as the observable variable, establishing the role assignment that distinguishes hidden states (spatial motion primitives) from observations (temporal progression).

The adjustable parameter `CB_groups` provides a direct pedagogical lever: students can observe how coarser discretizations produce simpler models at the cost of spatial resolution, while finer discretizations increase expressiveness at the risk of data sparsity.

3.6. HMM Training

With symbolic state and observation sequences in hand, the HMM transition matrix ($\mathbf{A}$), the emission matrix ($\mathbf{B}$), and the initial state distribution ($\boldsymbol{\pi}$) are estimated directly from the encoded training demonstrations. The number of hidden states and observable symbols is inferred from the encoded sequences, resulting in a fully data-driven model structure that requires no manual specification of the state space.

Transition probabilities are initialized uniformly and augmented with pseudotransitions to improve numerical stability and prevent zero-probability entries that would otherwise cause the training procedure to collapse. This implementation detail is made explicit in the LiveScript, connecting the theoretical requirement of a well-defined probability distribution to a concrete numerical safeguard that students would otherwise encounter silently inside a library.

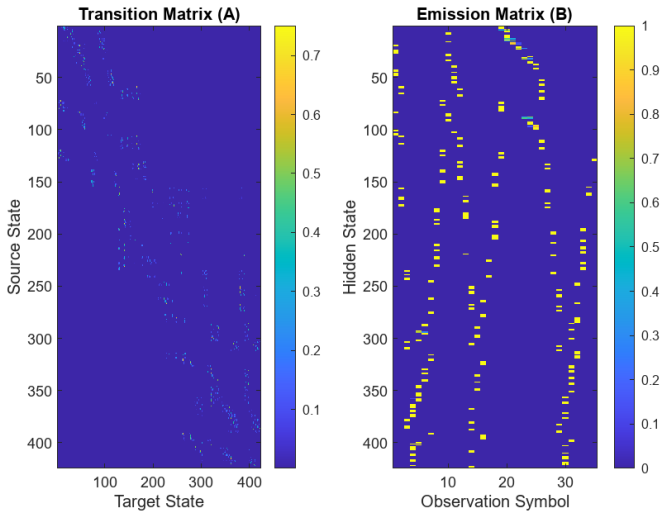

Figure 4: Learned HMM parameters visualized as heatmaps. Lighter cells indicate higher probability. The transition matrix encodes the sequential structure of motion primitives, while the emission matrix associates each hidden state with a discretized temporal observation.

Upon convergence, the learned matrices **A** and **B** are visualized as heatmaps, as shown in Figure 4. This visualization allows students to read the model's learned structure directly, identifying dominant state transitions and understanding which temporal observations are associated with which spatial motion primitives. The ability to interpret model parameters is a core competency that black-box implementations cannot develop.

3.7. Viterbi Decoding and Trajectory Reconstruction

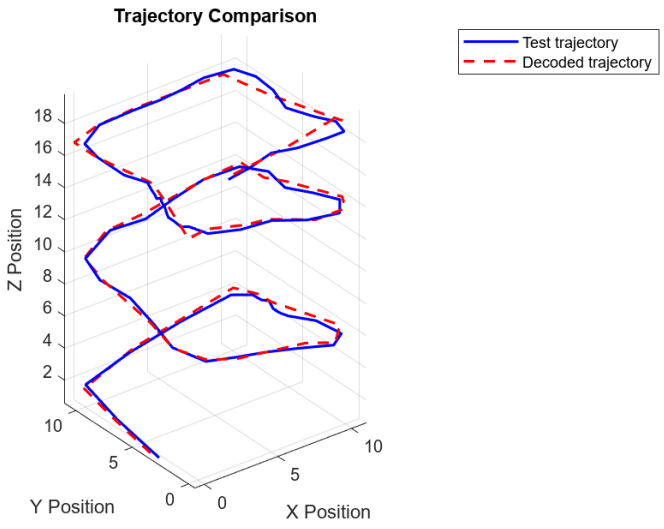

Figure 5: Comparison between the low-noise verification trajectory (solid) and the trajectory reconstructed by the trained HMM via Viterbi decoding (dashed). The visual proximity of the two curves provides students with an intuitive measure of model quality and closes the full continuous-to-symbolic-to-continuous learning loop.

The final stage closes the pedagogical loop by demonstrating the inference direction: given a new, unseen trajectory, the trained HMM infers the most probable sequence of hidden states using the Viterbi algorithm. The inferred state sequence is then decoded back into continuous spatial coordinates using the inverse normalization procedure (`normDecode.m`) and the global min–max statistics recorded during training.

The evaluation is performed on a low-noise verification trajectory generated with the same reference geometry but with noise amplitudes set significantly below the training level, emulating a high-quality execution. The reconstructed trajectory is plotted alongside the original in Figure 5, providing students with a direct, visual assessment of the model's generalization capability.

4. Educational Value and Discussion

4.1. Alignment with Engineering Education Principles

The pedagogical design of the tutorial is grounded in established principles from engineering education research, each of which maps directly onto structural decisions made in the LiveScript implementation.

Guided instruction. The pipeline progresses from concrete and controllable inputs toward increasingly abstract representations, culminating in a fully trained probabilistic model. At no point is a student required to engage with a concept before the prerequisites for that concept have been made explicit and executable. This scaffolded progression aligns with the recommendations of Rossiter et al. (2020), who argue that effective engineering courses must prioritize the gradual development of modeling intuition over front-loaded mathematical formalism.

Reduction of extraneous cognitive load. Sweller's Cognitive Load Theory identifies unnecessary cognitive load, such as the mental effort imposed by poor instructional design rather than by the intrinsic complexity of the material, as a primary obstacle to learning (Sweller, 1988). A common source of extraneous load in HMM instruction is the fragmentation of the learning experience across multiple tools: a textbook for theory, a separate programming environment for implementation, and external plotting utilities for visualization. The MATLAB LiveScript format eliminates this fragmentation by unifying executable code, numerical outputs, visualizations, and explanatory text within a single scrollable document (MathWorks, 2024a). Students maintain a continuous narrative thread through the pipeline, reducing the cognitive overhead of context–switching between environments.

Active learning through parameter exploration. Prince (2004) meta-analysis of active learning in engineering education establishes that student engagement with material produces measurably better conceptual retention than passive exposure. The tutorial operationalizes this principle through a set of explicitly exposed, semantically meaningful parameters: noise amplitudes (`np`, `nv`), simplification threshold (`threshold`), and codebook resolution (`CB_groups`). Each parameter has a predictable and visually observable effect on the pipeline output, inviting students to form hypotheses, modify values, and assess the consequences; a hands–on cycle that passive textbook study or black-box library usage cannot replicate (Bellas et al., 2024).

4.2. Transparency as a Pedagogical Strategy

A recurring theme across the six pipeline stages is the exposure of implementation details that are typically hidden inside software libraries. The pseudotransition initialization in the HMM training stage, the data–driven construction of the symbolic codebook, and the explicit inversion of the normalization during reconstruction are all made visible and executable. This transparency allows students to connect each implementation decision to a theoretical requirement, developing the diagnostic competency needed to apply and adapt HMMs to novel problems beyond the tutorial's scope.

5. Conclusions and Future Work

This paper has presented a publicly available MATLAB LiveScript tutorial for teaching Hidden Markov Models in the context of multi–dimensional trajectory learning. The tutorial addresses the absence of executable, transparent resources that guide students through the complete modeling pipeline within a single coherent learning experience.

The tutorial's design is grounded in the reduction of extraneous cognitive load, and active learning through parameter exploration. Each of the pipeline stages introduces one conceptual layer at a time, exposes its implementation explicitly, and produces a visually inspectable intermediate result. These stages form a round-trip from continuous trajectories through symbolic abstraction and back, making the purpose and internal structure of the HMM tangible.

The algorithmic foundations of the tutorial were previously introduced by the authors as a technical contribution to robotic trajectory learning (Manrique-Cordoba et al., 2025). The present work contributes the instructional design layer: the pedagogical sequencing, the parameterization strategy, and the contextualization of the pipeline within engineering curricula. The tutorial is publicly available on MATLAB Central File Exchange (Manrique-Cordoba, 2026), making it immediately deployable in undergraduate and graduate courses in artificial intelligence, robotics, and signal processing.

Future work will pursue empirical validation through formal classroom deployment, measuring conceptual gains, task completion, and student self-efficacy across different course contexts. Additionally, the tutorial will be extended with new datasets varying in geometric complexity and data dimensionality, tailored to specific course contexts, each accompanied by targeted questions designed to guide students through particular modeling challenges.

Agradecimientos

This research was funded by Spanish Government—Agencia Estatal de Investigación (AEI) through the project PID2022-138206OB-C32 and by the Generalitat Valenciana through the project CIPROM/2022/16.

References

Bellas, F., Naya-Varela, M., Mallo, A., et al., 2024. Education in the AI era: a long-term classroom technology based on intelligent robotics. Humanities and Social Sciences Communications 11, 1425. DOI: 10.1057/s41599-024-03953-y

Blunsom, P., 2004. Hidden Markov models. Lecture Notes 15 (18–19), 48.

Galli, M., Barber, R., Garrido, S., Moreno, L., 2017. Learning robotics with ROS and MATLAB using real robotic platforms. In: Proceedings of ICERI2017. IATED, pp. 1519–1527.

Manrique-Cordoba, J., 2026. MultiDimensionalData HMM Implementation Example. MATLAB Central File Exchange, https://es.mathworks.com/matlabcentral/fileexchange/183205-multidimensionaldatahmmimplementationexample, accessed: June 2026.

Manrique-Cordoba, J., de la Casa-Lillo, M. A., Sabater-Navarro, J. M., 2025. N-dimensional reduction algorithm for learning from demonstration path planning. Sensors 25 (7), 2145. DOI: 10.3390/s25072145

MathWorks, 2024a. Create live scripts in the live editor — MATLAB & Simulink. https://es.mathworks.com/help/matlab/matlab_prog/create-live-scripts.html.

MathWorks, 2024b. Hidden Markov models (HMM) — MATLAB & Simulink. https://es.mathworks.com/help/stats/hidden-markov-models-hmm.html.

Prince, M., 2004. Does active learning work? A review of the research. Journal of Engineering Education 93 (3), 223–231.

Rossiter, J. A., Serbezov, A., Visioli, A., Žáková, K., Huba, M., 2020. A survey of international views on a first course in systems and control for engineering undergraduates. IFAC Journal of Systems and Control 13, 100092.

Sweller, J., 1988. Cognitive load during problem solving: Effects on learning. Cognitive Science 12 (2), 257–285.