LATMOS: Latent Automaton Task Model from Observation Sequences

Weixiao Zhan[1] Qiyue Dong[2] Eduardo Sebastián[3] Nikolay Atanasov[2]

***Abstract*— Robot task planning from high-level instructions is an important step towards deploying fully autonomous robot systems in the service sector. Three key aspects of robot task planning present challenges yet to be resolved simultaneously, namely, (i) factorization of complex tasks specifications into simpler executable subtasks, (ii) understanding of the current task state from raw observations, and (iii) planning and verification of task executions. To address these challenges, we propose LATMOS, an automata-theory-inspired task model that, given observations from correct task executions, is able to factorize the task, while supporting verification and planning operations. LATMOS combines an observation encoder to extract features from potentially high-dimensional observations with a sequence model that encapsulates an automaton with symbols in the latent feature space. We conduct evaluations in three task model learning setups: (i) abstract tasks described by logical formulas, (ii) real-world human tasks described by videos and natural language prompts and (iii) a robot task described by image and state observations. The results show improved plan generation and verification capabilities of LATMOS across different observation modalities and tasks.**

I. INTRODUCTION

Robot systems are increasingly integrated in unstructured human environments and expected to perform increasingly more complicated tasks autonomously. An important consideration is how to describe the objectives and constraints of a desired task to a robot, enabling human-understandable high-level specifications to be the main signal to command the robot systems. Task specification refers to the formal description of goals, constraints, and execution dependencies associated to a task, and it serves as a fundamental component of formulating autonomous robot behaviors, enabling robots to understand, decompose, and execute complex objectives while maintaining safety and failure recovery.

One popular formalism for expressing task specifications is Linear Temporal Logic (LTL) [1]–[4]. Every LTL formula is equivalent to a Büchi automaton [5], [6], a class of automata that admits infinite input sequences and plays an important role in model checking [7], [8]. Automata-based task specification methods rely on explicit *pre-defined* symbolic representation — logical propositions associated to certain events for the system — and automaton states and transitions, leading to interpretable graphical models that can be used either for task verification [9] or planning [10], [11]. However, constructing an automaton that represents a task requires substantial manual engineering effort and often restricts the symbols and task states to discrete spaces that lack the expressivity to capture real-world robotics scenarios involving high-dimensional image data and continuous robot states.

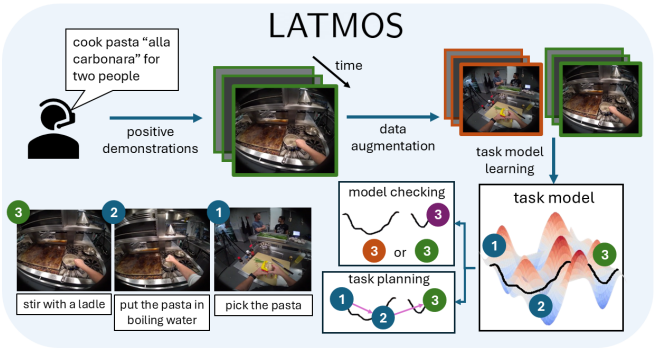

Fig. 1: We present LATMOS, an automata-theory-inspired method to learn continuous task models from raw observations of positive task demonstrations. Given a dataset of task executions, our method first synthesizes negative demonstrations. Then, an encoder→sequence-model→decoder architecture learns a continuous latent-space representation of a task automaton. After training, the model is used for task plan verification and generation.

On the other hand, recent theoretical advances have established principled connections between Recurrent Neural Networks (RNNs) and Deterministic Finite Automata (DFA) [12], suggesting the potential for learning continuous task representations. Concurrently, the success of pre-trained feature extractors [13]–[15] offers the ability to capture complex patterns and relationships in continuous embedding spaces from images, videos and natural languages. However, there is still a gap to connect these theoretical results and practical insights to address robot task representation, in particular, in how to design a training signal from expert data to learn a meaningful automata-inspired task representation. Consequently, an important question is: *given observations of task demonstrations, can we learn a continuous task representation that allows to validate a robot's execution of the task and plan ahead the next steps in the task space?*

We formalize this question in Sec. III and propose an approach, termed as LATMOS, for learning latent task models from positive task demonstrations in Sec. IV. Our approach is illustrated in Fig. 1. Instead of relying on

We gratefully acknowledge support from ARL DCIST CRA W911NF-17-2-0181, ONR N00014-23-1-2353, and a US-Spain Fulbright grant.

[1]W. Zhan is with the Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA 92093, USA (e-mail: wezhan@ucsd.edu).

[2]Q. Dong and N. Atanasov are with the Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093, USA (e-mail: {q4dong, natanasov}@ucsd.edu).

[3]E. Sebastián is with the Department of Computer Science and Technology, University of Cambridge, Cambridge, CB3 0FD, UK (e-mail: es2121@cam.ac.uk).

handcrafted symbols and task states, LATMOS leverages multi-modal encoders to directly extract symbols from the observations. Then, a sequence model learns an internal state representation in a continuous domain that implicitly encodes the steps of the task. Lastly, a decoder model transforms the task state to a human interpretable signal. In Sec. V, we demonstrate that LATMOS achieves superior performance in model checking and robot task planning compared to automata-based approaches across different environments and tasks using simulated and real data. These results show that LATMOS can be integrated effectively as a high-level decision-making component in the stack of autonomous robot systems.

II. Related Work

Propositional logic [16], temporal logic [17], and automata [18] provide a structured formulation of task specifications for robot systems. Since automata are graphical representations, they can be composed with the graph structures used in robot motion planning algorithms to achieve task planning [19]–[22]. Temporal logic specifications [23], [24] capture the temporal ordering of events and can be converted to equivalent automata, enabling optimal control of long-horizon tasks in high-dimensional robot systems. Recent works incorporate semantic reasoning and natural language understanding into temporal logic task planners [25]–[28], allowing robots to interpret high-level commands in unstructured environments. While temporal logic and automata models have a clear structure amenable to providing task execution guarantees, they rely on handcrafted specifications of propositions and task states. Our approach bypasses this by integrating learning with automata principles, allowing robots to *learn* task structure, monitor execution states, and plan task transitions.

Traditional automata learning methods, like Angluin's L* algorithm [29] and its extensions [30], are based on a tabular classification of sequences of symbols and require positive demonstrations and negative counterexamples from an oracle. Statistical approaches like ALERGIA [31] overcome the need of counterexamples, using positive sequences of symbols to compute the transition probabilities among states of a stochastic finite automaton (SFA). Spectral methods [32]–[34] conduct a frequency analysis over the symbols to infer both the transition probabilities and the states of an SFA, enabling model verification and planning by selecting and comparing the transition probability given a state. Acknowledging the need for more flexible approaches, Li et al. [12] theoretically connect weighted automata, tensor networks, and recurrent neural networks via spectral learning, a connection that has not been fully exploited yet. Baert et al. [35] use an SFA to model task execution and, subsequently, learn the model from demonstrations. In the robot task planning domain, Araki et al. [10], [18] use automata as an internal representation of a "logical" Markov Decision Process that is called in a Value Iteration process for planning. Although these approaches exploit learning to add flexibility to automata-based task representations, they still rely on handcrafted symbols or other classes of logical templates. Instead, one of the focus of our work is to avoid manual engineering efforts and develop a method that automatically learns task models directly from observations.

Recently, works on task planning based on Large Language Models (LLMs) demonstrate their ability to generate action sequences directly from observations and prompts without explicit task structures [36]–[38]. While such approaches enable flexible planning, they often lack correctness guarantees due to their implicit reasoning. This limitation is mitigated by works that use world models [39] or hierarchical scene graphs [40] to improve LLM-based planning. An alternative paradigm is to decompose task planning into explicit task, robot, and environment models, allowing for modularity, verifiable correctness and reliability. We propose LATMOS to learn a task model from observations that enables plan generation and verification in the task space rather than the action space, decoupling the task model from the robot model and action policy.

III. Problem Definition

This section introduces the definitions and data needed to formalize the problem of task model learning. Sec. III-A presents background on automata theory, Sec. III-B introduces the main features of the data assumed to be available, and Sec. III-C formalizes the problem.

A. Deterministic Finite Automata

A task specification can be encoded as a Deterministic Finite Automaton (DFA) operating in a discrete time $k \in \mathbb{N}$. A DFA is a tuple $\mathcal{A} = (S, \Sigma, \delta, s_0, F)$, where:

- S is a finite set of states,
- Σ is a finite set of inputs, called alphabet,
- $\delta : S \times \Sigma \to S$ is a transition function such that the next state is given by a transition $s_{k+1} = \delta(s_k, o_k)$ dependent on the current state $s_k \in S$ and input $o_k \in \Sigma$,
- $s_0 \in S$ is an initial state,
- $F \subseteq S$ is a set of accepting (final) states, which model the completion of a task.

We assume that the automaton $\mathcal{A}$ representing the task, including all of its elements S, Σ, δ, s_0 and F, is unknown.

B. Positive Task Demonstrations

Instead of an automaton representation, we assume that the task of interest is described by observation sequences from successful task demonstrations. Specifically, let $o_k^j \in \mathbb{R}^N$ be an N-dimensional observation from task demonstration j at time k. The sequence of observations obtained from one task execution is denoted by $o^j = \{o_k^j\}_{k=0}^{K_j}$, where $K_j > 1$ is the time-horizon of the demonstration. The collection of observations from all demonstrations is denoted by $O = \{o^j\}_{j=1}^{L}$, where L is the number of demonstrations.

We emphasize that we assume the availability of positive demonstrations only. In general robotics applications, the space of unsatisfactory task executions is much larger than that of the positive demonstrations. Furthermore, negative demonstrations of a robotic task, such as assistance in a

hospital setting, involves concerns and risks that cannot be taken. Therefore, in a safety-critical context, it is reasonable to assume the absence of negative samples in O.

C. Problem Definition

Our objective is to learn a task model with *continuous* states given *continuous* observations O from positive task demonstrations. The purpose of learning a continuous model instead of a discrete (automaton) model is to enable flexibility with respect to novel observations out of the distribution of the positive demonstrations contained in O, and robustness against perturbations that arise in unstructured environments typical of robotic applications. We also consider continuous states in the task model to allow learning a suitable latent state space and state transition function. Let $\mathcal{M}_\theta(o^j)$ be a neural network model parameterized by $\theta \in \mathbb{R}^M$ which takes an observation sequence o^j as input. We consider the following problem.

Problem 1. Given positive task demonstrations O, find parameters θ^* such that the model $\mathcal{M}_\theta(o^j)$ is a continuous representation of an automaton $\mathcal{A}$ with the following capabilities.

1) **Model checking**: Given an observation sequence o^j, model $\mathcal{M}_{\theta^*}$ can verify if o^j satisfies the task.
2) **Task planning**: Given an observation sequence o^j, model $\mathcal{M}_{\theta^*}$ can predict an extension $\tilde{o}^j$ such that the concatenated sequence $o^j||\tilde{o}^j$ satisfies the task.

Having access to a task model with the properties in Problem 1 would allow a robot system to plan a task execution and verify its correctness relying on sensor observations.

IV. Learning Task Representations for Planning

To solve Problem 1, we propose LATMOS (Latent Automaton Task Model from Observation Sequences), a method that combines the benefits of a neural network encoder to extract features from high-dimensional observations with a neural network sequence model to represent the task structure, and a neural network decoder to output the probability that an observation sequence comes from a correct task execution. Fig. 2 presents an overview of LATMOS. In Sec. IV-A, we describe how to learn task models from just positive task demonstrations. In Sec. IV-B, we describe how to use a neural network encoder to overcome the need for handcrafted discrete input symbols as in an automaton representation. In Sec. IV-C, we derive a task planning method based on inference-time optimization, enabling task planning in the learned latent task space.

A. Training Task Models Through Model Checking

The first challenge of learning task representations solely from positive demonstrations is that the training data is inherently unbalanced. To tackle this challenge, we augment the observation dataset O with synthetically generated negative examples. For ease of exposition, we first assume that the set of observations O is formed by sequences from a known automaton alphabet Σ. Later on, in Sec. IV-B, we describe

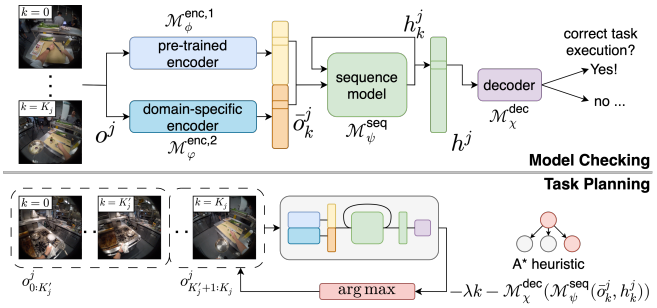

Fig. 2: Overview of LATMOS. **(Top)** Given observation sequences from task demonstrations (e.g., cooking videos), a neural network encoder is used to extract features from each observation. The resulting feature sequences are used to train a sequence model and a decoder with a model-checking objective to solve the first item of Problem 1. **(Bottom)** Once trained, the task model may be queried to obtain a heuristic function as guidance for a motion planning algorithm executed in the learned latent task space.

how to deal with high-dimensional observations (e.g., images) and remove the assumption of a known alphabet.

For each sequence of observations o^j, we select an observation o^j_k, copy the prior observation sequence $o^j_{0:k}$ and denote it by $\tilde{o}^j_{0:k}$. Then, we sample a sequence of new synthetic observations $\tilde{o}^j_{k+1:K_j}$ uniformly and independently from the alphabet Σ. The sequences $\tilde{o}^j_{0:k}$ and $\tilde{o}^j_{k+1:K_j}$ are concatenated, leading to a synthetic negative demonstration $\tilde{o}^j$. This step is repeated for each $o^j_k \in o^j$. The synthetic negative demonstrations are included in a augmented dataset, denoted as $\tilde{O}$, which also contains the original positive demonstrations in O. After that, binary labels are assigned to the demonstrations. Label $y^j_k = 1$ is assigned to all the observations o^j_k that belong to the original dataset O and those belonging to the prior sequences of observations $o^j_{0:k}$ sampled to generate the negative synthetic sequence $\tilde{o}^j$. Label $y^j_k = 0$ is assigned to all the new synthetic observations $\tilde{o}^j_{k+1:K_j}$. This method produces a balanced dataset with observations from positive and negative task executions. Our synthetic label generation approach is feasible under the assumption that the space of negative demonstrations is exponentially larger than the space of positive demonstrations, such that most random observation sequences lead to an incorrect execution of the task. In practice, this assumption holds when the observation dimension N is very large, e.g., in the case of robot task executions with visual observations.

Given the augmented dataset $\tilde{O}$, we pose the problem of learning $\mathcal{M}_{\theta^*}$ as the minimization of the cross-entropy $H(\hat{y}^j_k, y^j_k)$ between the prediction of the model $\hat{y}^j_k$ and the ground-truth labels y^j_k associated to $\tilde{O}$. We use one-hot encoding on the binary ground-truth labels y^j_k. To obtain $\hat{y}^j_k$, we parameterize $\mathcal{M}_\theta$ by means of two neural networks. The first neural network is a sequence model $\mathcal{M}^{\text{seq}}_\psi(o^j_k, h^j_k)$, parameterized by ψ, that is in charge of learning a latent state representation of the automaton $\mathcal{A}$, denoted as h^j_k. The

second neural network is a decoder $\mathcal{M}_{\chi}^{\mathsf{dec}}(h_{K_j}^j)$, parameterized by χ, that takes the latent state h_k^j as input and outputs the probability of acceptance of the sequence of observation $o_{0:k}^j$. The activation function of the last layer of the decoder is set to be SoftMax to obtain a vector $\hat{y}_k^j$ containing the probability of acceptance and failure.

Learning a continuous representation of the task from demonstrations then translates into finding parameters ψ and χ that minimize the cross-entropy loss between ground-truth labels and predicted labels:

$$\begin{aligned} \min_{\psi,\chi} \quad & \mathbb{E}_{o_k^j \sim \tilde{O}} \left[\frac{1}{L} \sum_{j=1}^{L} \frac{1}{K_j} \sum_{k=0}^{K_j} H\left(\hat{y}_k^j, y_k^j\right) \right] \\ s.t. \quad & h_{k+1}^j = \mathcal{M}_{\psi}^{\mathsf{seq}}(o_k^j, h_k^j) \quad \forall k, j, \\ & \hat{y}_k^j = \mathcal{M}_{\chi}^{\mathsf{dec}}(h_k^j) \quad \forall k, j. \end{aligned} \tag{1}$$

In the next section, we extend the formulation to handle high-dimensional observations from an unknown Σ.

B. From Symbols to High-Dimensional Observations

Task planning in real-world robotics applications cannot rely on hand-crafted finite alphabets because it is hard to choose a set of discrete symbols that describes all possible variations of high-dimensional observations, such as images from an onboard camera. We propose an encoder-based method to automatically infer relevant symbols from task demonstrations in the form of latent-space features.

Let o_k^j be a high-dimensional observation of the environment at time k from demonstration j, e.g., an image or a set of frames grouped together in the same vector. Instead of directly using o_k^j as input to $\mathcal{M}_{\psi}^{\mathsf{seq}}$, we use a neural network encoder to extract features from o_k^j. Specifically, the encoder is a combination of a pre-trained model $\mathcal{M}_{\phi}^{\mathsf{enc},1}(o_k^j)$ with parameters ϕ, and a small task-specific encoder $\mathcal{M}_{\varphi}^{\mathsf{enc},2}(o_k^j)$ with parameters φ. The outputs of the two models are concatenated to obtain $\bar{o}_k^j$. The pre-trained encoder maps semantically similar inputs to nearby regions in the embedding space, making extracted features behave similarly to symbols of an automaton alphabet. Hence, ϕ is frozen during training. On the other hand, the task-specific encoder is a smaller model that is jointly trained with the sequence model and decoder to adapt domain-specific observations.

Using the feature encoders, minimization (1) is reformulated as follows:

$$\begin{aligned} \min_{\psi,\chi,\varphi} \quad & \mathbb{E}_{o_k^j \sim \tilde{O}} \left[\frac{1}{L} \sum_{j=1}^{L} \frac{1}{K_j} \sum_{k=0}^{K_j} H\left(\hat{y}_k^j, y_k^j\right) \right] \\ s.t. \quad & \bar{o}_k^j = \mathcal{M}_{\phi}^{\mathsf{enc},1}(o_k^j) || \mathcal{M}_{\varphi}^{\mathsf{enc},2}(o_k^j) \quad \forall k, j, \\ & h_{k+1}^j = \mathcal{M}_{\psi}^{\mathsf{seq}}(\bar{o}_k^j, h_k^j) \quad \forall k, j, \\ & \hat{y}_k^j = \mathcal{M}_{\chi}^{\mathsf{dec}}(h_k^j) \quad \forall k, j. \end{aligned} \tag{2}$$

In (2), the outputs of the encoder are interpreted as input symbols of an automatically inferred alphabet with N' symbols. Unlike the symbols in a DFA, our model treats the embedding vectors as continuous symbols, which enables robustness against the variability of the real-world situations that robots encounter during deployment, e.g., different camera perspectives and object rearrangements.

In summary, the proposed model $\mathcal{M}_{\theta}(o^j)$ to solve Problem 1 is composed of three modules: (i) an encoder $(\mathcal{M}_{\phi}^{\mathsf{enc},1}(o_k^j), \mathcal{M}_{\varphi}^{\mathsf{enc},2}(o_k^j))$ that takes high-dimensional observations and learns to automatically extract the symbols of a relevant alphabet; (ii) a sequence model $(\mathcal{M}_{\psi}^{\mathsf{seq}}(\bar{o}_k^j, h_k^j))$ that learns a latent state representation of the task; and (iii) a decoder $(\mathcal{M}_{\chi}^{\mathsf{dec}}(h_{K_j}^j))$ that learns to interpret the latent state of the sequence model and assess whether the task is solved. The trainable model parameters are $\theta = \{\varphi, \psi, \chi\}$. By training the model $\mathcal{M}_{\theta}(o^j)$ as in (2), we address the model checking requirement in item 1 of Problem 1. We discuss how to use $\mathcal{M}_{\theta}(o^j)$ for task planning to address item 2 of Problem 1 next.

C. Latent Space Task Planning

LATMOS enables task planning through test-time optimization. Given an observation sequence $o_{0:K_{j'}}^j$ and our model $\mathcal{M}_{\theta}(o^j)$, we pose task planning as the following optimization problem:

$$\begin{aligned} \underset{\bar{o}_{K_{j'}+1:K_{j''}}^j}{\arg\max} \quad & \mathcal{M}_{\chi}^{\mathsf{dec}}(h_{K_{j''}}^j) \\ s.t. \quad & h_{k+1}^j = \mathcal{M}_{\psi}^{\mathsf{seq}}(\bar{o}_k^j, h_k^j) \; \forall k \in \{K_j', \ldots, K_j''\}, \end{aligned} \tag{3}$$

with $\bar{o}_{K_{j'}+1:K_{j''}}^j = \{\bar{o}_k^j\}_{k=K_{j'}}^{K_{j''}}$ and $K_j'' > K_j'$ the last planning step. To solve (3), we use a model-guided A* search [41]. To apply A*, we assume that we have access to a simulator or an observation model that can be queried to synthesize task paths (see, e.g., the 2D grid environment in Sec. V-C). Given an initial observation $\bar{o}_{K_{j'}}^j$ and associated latent hidden state $h_{K_{j'}}^j$, we define the following heuristic function:

$$\gamma(h_k^j, k) = -\lambda k - \mathcal{M}_{\chi}^{\mathsf{dec}}(\mathcal{M}_{\psi}^{\mathsf{seq}}(\bar{o}_k^j, h_k^j)), \tag{4}$$

where λ trades off search exploration and current task path extension. The dynamic programming nature of A* allows efficient exploration of a space with exponentially many path by reusing previously computed latent hidden states. The output of the model-guided A* algorithm is a sequence of observations and associated latent states that solve the task. The observation sequence can then be used as input to a robot control algorithm to generate actions that lead to observations aligned with the desired observation sequence.

Alternatively, since our task model is differentiable, it is possible to employ gradient-based optimization for task planning. As the exact time-horizon of the plan is not known a priori, one can set a pre-defined horizon T and do planning in the latent task space in a receding-horizon model predictive control fashion, leveraging $\mathcal{M}_{\theta}(o^j)$ as a dynamic model of the task. We leave the analysis of this approach for future work. Both planning methods enable robots to exploit model $\mathcal{M}_{\theta}(o^j)$ for task planning, even though it is trained using just a model checking loss from

positive demonstrations. Hence, LATMOS addresses item 2 in Problem 1.

V. EXPERIMENTS

We evaluate LATMOS in three types of experiments[1]:

1) synthetic tasks specified by LTL formulas using Spot [42], which allow to assess model checking with different sequence model configurations of LATMOS under known alphabets in Sec. IV-A;
2) real-world tasks specified by video demonstrations using the Ego-Exo4D dataset [43], which allow to asses the model checking properties of LATMOS when the alphabet is unknown and only raw high-dimensional observations are available in Sec. IV-B;
3) goal-conditioned tasks in a Door-Key grid environment [44], which allows to assess the domain-specific encoder and task planning of LATMOS in Sec. IV-C.

In all experiments, the decoder is parameterized as a Multi-Layer Perceptron (MLP) with one hidden layer whose dimension is the average of the input and output dimension. The MLP uses Leaky ReLU activations in the hidden layer. The decoder is deliberately simple to force the sequence model to learn a high-quality latent representation.

Since model checking with LATMOS reduces to binary classification of an input sequence O^j, the evaluation metric for model checking is acceptance accuracy,

$$\text{acceptance accuracy} = \frac{\text{TP}_j + \text{TN}_j}{\text{TP}_j + \text{TN}_j + \text{FP}_j + \text{FN}_j},$$

where TP, TN, FP and FN refer to the number of true positives, true negatives, false positives and false negatives respectively. Regarding the task planning capabilities of LATMOS, we consider search efficiency as the evaluation metric:

$$\text{search efficiency} = \frac{E_j}{{K_j}''},$$

where E_j is the number of explored states and ${K_j}''$ is the length of the shortest task plan achieved. For both metrics, we compute the average over the test set sequences.

A. Automata Learning from Positive Demonstrations

We first evaluate our method on synthetic tasks specified by randomly generated LTL formulas with corresponding ground-truth automata. To generate training and test datasets, we use 3 automata (see Fig. 3) and generate $L = 1000$ sequences from each via random walks on the automaton graphs, with varying sequence lengths up to $K_j = |S|$ to ensure enough exploration. Since we have access to the ground-truth automata, dataset augmentation is not needed and we directly employ the Atomic Propositions (APs) of the LTL formulas as observations. Three variations of the same LTL formula are tested:

- **Base**: Training and testing use exact APs collected by the random walks.

[1]Code at: https://github.com/weixiao-zhan/LATMOS

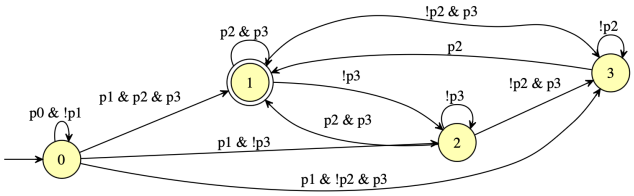

Fig. 3: Example of ground-truth automaton with 4 APs and 4 states. State 1 is the accepting state of the task.

- **Noisy APs**: Training APs are perturbed by additive zero-mean Gaussian noise with covariance $\mathbf{Q} = \{0.1\mathbf{I}, 0.2\mathbf{I}\}$, where $\mathbf{I}$ is the identity matrix of appropriate dimensions. Testing APs are kept exact.
- **Novel APs**: The test sequences contain transitions absent in the training dataset.

We compare LATMOS against two traditional automata learning methods: ALERGIA [45] and Spectral Learning (Sp-Learn) [32]. ALERGIA builds a look-up table during training and traverses the table to assess acceptance at test time. Sp-Learn constructs a Hankel matrix to capture the frequency of chains of APs and to learn the states S and the transition function δ. For Sp-Learn, the rank factor is the ratio between the number of states in the learned automaton, i.e., the rank of the Hankel matrix, and the actual number of states in the ground-truth automaton. Regarding LATMOS, we ablate three most popular parametrizations of the sequence model $\mathcal{M}_\psi^{\text{seq}}$: a Gated Recurrent Unit (GRU) [46] network, an Attention-based network [47], and a State-Space Model (SSM) [48] network. All parametrizations are configured to have similar number of parameters and trained with the same number of epochs and learning rate. The *hidden factor* is the ratio between the dimension of the hidden state and the number of states in the ground-truth automaton.

Tables I and II show the acceptance accuracy for the different setups and model configurations. Sp-Learn performs the best when its Hankel Matrix has exactly the same rank as the number of states in ground-truth automata, which implies that it is a critical factor but it may be hard to tune in the absence of ground-truth automata. In contrast, LATMOS performs consistently with large-enough hidden dimensions. We note that an Attention network with a hidden factor of 0.5 may not be implemented due to model-size incompatibilities inherent to the multi-head attention architecture. On the base dataset (Table I), all models achieve high accuracy, confirming their ability to learn automata. However, in the presence of noisy or novel APs (Table II), ALERGIA, Sp-Learn completely fail due to their reliance on exact AP matching. On the other hand, LATMOS performs consistently in both cases. Specifically, the Attention backbone slightly outperforms GRU and SSM. This is deemed to the fact that Attention processes the whole observation sequence at each step; meanwhile, GRU and SSM are recurrent methods that process information step by step. These results validate that

TABLE I: Accuracy for the base configuration.

model configuration	rank/hidden factor			
	0.5	1	4	12
ALERGIA [45]		**1.000**		
Sp-Learn [32]	0.834	**0.948**	0.945	0.925
LATMOS (GRU)	0.125	0.314	0.720	**0.957**
LATMOS (Attention)	-	0.148	0.956	**1.000**
LATMOS (SSM)	0.125	0.520	0.895	**0.956**

TABLE II: Accuracy for the noisy and novel configurations.

model configuration	rank/hidden factor	accuracy		
		var=0.1	var=0.2	novel
ALERGIA [45]	-	0	0	0
Sp-Learn [32]	1	0	0	0.639
LATMOS (GRU)	12	0.775	0.668	0.686
LATMOS (Attention)	12	**0.794**	**0.681**	**0.864**
LATMOS (SSM)	12	0.787	0.666	0.648

our learned latent representations capture automata structure while making meaningful interpolations and generalizing to noisy and unseen APs, which are critical properties for real-world robot applications where observations are imperfect.

B. Automata Learning from Real-World Visual Cues

Ego-Exo4D [43] is a diverse, large-scale multi-modal, multi-view video dataset that contains time-aligned egocentric and exocentric videos of people completing tasks like cooking or repairing a bike. Specifically, we use the key-step sub-dataset, in which the segments of the videos are annotated with step ID and natural-language captions. Sequences of segments constitute positive task demonstrations. In this experiment, three state-of-the-art pre-trained feature extractors are compared: XCLIP [15], Omnivore [13], and EgoVLPv2 [14]. The feature extractors are trained to handle Vision (V) or Language (L) modalities. XCLIP and Omnivore are trained in generalist datasets whereas EgoVLPv2 is specialized to Ego-Exo4D data. No task-specific encoder is used. For the data augmentation process, we leverage the encoder structure of LATMOS to directly sample in the embedding space. Following the insights gained from Sec. V-A, $\mathcal{M}_\psi^{\text{seq}}$ is set as an Attention-based model with 4 layers and 4 attention heads, with 512 hidden dimensions.

To further assess the benefits of LATMOS, we train a second key-step decoder to solve the single key-step recognition challenge included in Ego-Exo4D [43]. The goal of the challenge is to classify step IDs s_k^j from the visual observation segments o_k^j. We adopted LATMOS for this challenge to study whether the learned latent states h_k^j are representative of the task and help in improving prediction accuracy in the benchmark. The key-step decoder is parameterized as the original acceptance decoder except the output uses SoftMax activation with 629 dimension (the number of unique step IDs). We first train the sequence model with the acceptance decoder, as proposed in (1) and (2); then, we train the key-step decoder with $\mathcal{M}_\psi^{\text{seq}}$ frozen.

Table III summarizes the obtained results, with symbol † marking that the accuracy is reproduced from [14]. The first conclusion is, in terms of acceptance accuracy, LATMOS achieves consistent accurate performance across pre-trained encoders and high-dimensional observation modalities. This is specially relevant because it highlights the modularity LATMOS with respect to modules employed to extract relevant symbols from observations. On the other hand, for the top 2 best vision encoders (Omnivore and EgoVLPv2), the use of the latent representation of LATMOS significantly improves accuracy in the key-step challenge. We claim that the boost is due to the fact that having a meaningful latent representation of the task helps discerning between observations that very close in the observation space but have quite different semantic meaning and step IDs. An instance can be seen in Fig. 4, where two images that are almost equal have different captions.

(a) Caption: "add salt" (b) Caption: "add sugar"

Fig. 4: Example of key steps that have similar video frames but different step ID and language annotations.

TABLE III: Acceptance accuracy and key-step accuracy in the Ego-Exo4D dataset for different pre-trained encoders.

encoder	modality	acceptance	key-step	
		LATMOS	single	LATMOS
XCLIP [15]	L	0.901	0.865	0.852
	V	0.681	0.353	0.387
	VL	0.902	0.806	0.626
Omnivore [13]	V	0.907	0.352	0.464
EgoVLPv2[14]	V	0.866	0.368†	0.448

C. Robot Task Planning from Learned Automaton

After evaluating the model checking capabilities of LATMOS, we assess its task planning capabilities in a 2D grid environment called Door-Key [44]. In this task, a robot must find a key to open a door and reach a goal while avoiding obstacles. We generate training data from 36 procedurally generated 8×8 grid environments with varying key, door, and goal positions. We consider two variants of LATMOS: LATMOS-x, which takes in explicit agent location, key location, goal location and door status; and LATMOS-v, which takes in bird-eye view RGB images of the environment (right part of Fig. 6). The domain-specific encoder of LATMOS-v is a one-layer five-channel convolutional neural network. No encoder is used for LATMOS-x, we directly feed the observation to the sequence model.

For task planning, we compare four heuristics for the model-guided A* planner: a Dijkstra variant corresponding

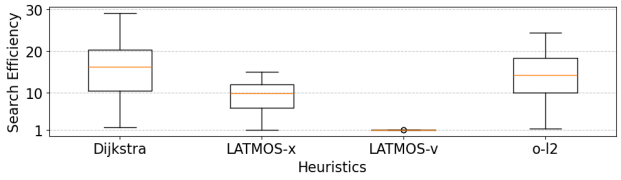

Fig. 5: Search efficiency for different heuristics in the Door-Key task. LATMOS significantly outperforms the other heuristics, almost always achieving a successful task plan by building a single-branch tree, i.e, $E \simeq K''$.

to a zero heuristic; the LATMOS-x heuristic as proposed in (4); the LATMOS-v heuristic as proposed in (4); and a o-l2 heuristic, which is the l_2 heuristic applied to the bird-eye images of the environment.

As shown in Fig. 5, the two LATMOS heuristics significantly outperform the baseline heuristics. In particular, LATMOS-v often computes a successful task plan by building a single-branch tree, as the ratio between number of explored states and the length of the shortest computed path is near 1. The better performance of the LATMOS heuristics underlines the fact that our proposed method is able to learn meaningful latent task representations not only for model checking, but also for enhanced planning. The improvement of LATMOS-v over LATMOS-x is because the domain-specific encoder learns embeddings that help with the planning, in contrast to directly learning the arithmetic relationships between robot location, key location, goal location and door status as in LATMOS-x. To better understand the planning features of LATMOS, we provide a qualitative visualization of task plans generated by the Dijkstra and LATMOS-v heuristics in Fig. 6. Although A* generates a valid task plan with both heuristics, LATMOS-v is significantly more efficient than Dijkstra, in accordance to the search efficiencies reported in Fig. 5.

VI. CONCLUSION

This paper proposed LATMOS, an automata-theory-inspired approach for learning task models from positive task demonstrations, which can subsequently be used for robot task planning. By integrating a neural network observation encoder with a latent-space sequence model, LATMOS effectively factorizes a task, interprets its execution state, and enables task planning in a continuous latent space. Our experiments show LATMOS's ability to generalize across various domains, from LTL-specified tasks to real-world video-based task demonstration to robot task planning. Our results indicate that LATMOS is capable of both traditional model checking and efficient task planning using the sequence model for heuristic guidance. Future work will focus on enhancing the interpretability of the latent feature space. This includes exploring feature space alignment techniques with language features, which could enable interpretation and task guidance from large language models (LLMs). Additionally, while our planning method relies on A* in the task space, we recognize the potential for integrating

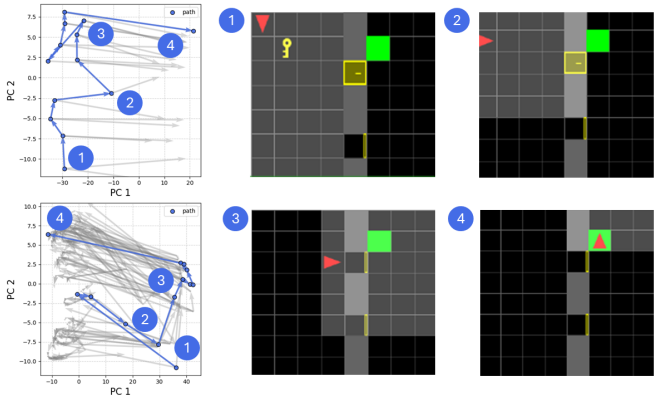

Fig. 6: Visualization of the task plans computed by LATMOS-v and Dijkstra. The panels on the left report PCA-projected *latent* state representations of the search tree produced by both heuristics (LATMOS-v on the top, Dijkstra on the bottom), in which, the blue trajectories are the final plan and the gray traces are A* search branches. The panels on the right display Door-Key configurations corresponding to four key states on the final plan (robot in red, goal in green, door in yellow, walls in gray). Both heuristics produced *the same optimal plans from same starting states*, but LATMOS-v is significantly better in search efficiency.

energy-based models [49] and optimization-based planning to avoid negative sampling and provide continuous-action planning. Finally, we plan to test LATMOS in real-world robot experiments.

REFERENCES

[1] M. Y. Vardi, "An automata-theoretic approach to linear temporal logic," *Logics for Concurrency: Structure versus Automata*, pp. 238–266, 2005.

[2] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Where's Waldo? sensor-based temporal logic motion planning," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 3116–3121.

[3] ——, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[4] Y. Kantaros, S. Kalluraya, Q. Jin, and G. J. Pappas, "Perception-based temporal logic planning in uncertain semantic maps," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2536–2556, 2022.

[5] M. Mukund, "Finite-state automata on infinite inputs," in *Modern Applications of Automata Theory*. World Scientific, 2012, pp. 45–78.

[6] S. L. Smith, J. Tumova, C. Belta, and D. Rus, "Optimal path planning under temporal logic constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 3288–3293.

[7] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[8] D. Kamale, E. Karyofylli, and C.-I. Vasile, "Automata-based optimal planning with relaxed specifications," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021, pp. 6525–6530.

[9] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, and A. Knoll, "A review of safe reinforcement learning: Methods, theories and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[10] B. Araki, K. Vodrahalli, T. Leech, C.-I. Vasile, M. Donahue, and D. Rus, "Learning and planning with logical automata," *Autonomous Robots*, vol. 45, no. 7, pp. 1013–1028, 2021.

[11] Z. Dai, A. Asgharivaskasi, T. Duong, S. Lin, M.-E. Tzes, G. Pappas, and N. Atanasov, "Optimal scene graph planning with large language model guidance," in *IEEE International Conference on Robotics and Automation*, 2024, pp. 14 062–14 069.

[12] T. Li, D. Precup, and G. Rabusseau, "Connecting weighted automata, tensor networks and recurrent neural networks through spectral learning," *Machine Learning*, vol. 113, no. 5, pp. 2619–2653, 2024.

[13] R. Girdhar, M. Singh, N. Ravi, L. Van Der Maaten, A. Joulin, and I. Misra, "Omnivore: A single model for many visual modalities," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 102–16 112.

[14] S. Pramanick, Y. Song, S. Nag, K. Q. Lin, H. Shah, M. Z. Shou, R. Chellappa, and P. Zhang, "Egovlpv2: Egocentric video-language pre-training with fusion in the backbone," in *IEEE/CVF International Conference on Computer Vision*, 2023, pp. 5285–5297.

[15] B. Ni, H. Peng, M. Chen, S. Zhang, G. Meng, J. Fu, S. Xiang, and H. Ling, "Expanding language-image pretrained models for general video recognition," in *European Conference on Computer Vision*. Springer, 2022, pp. 1–18.

[16] D. Meli, H. Nakawala, and P. Fiorini, "Logic programming for deliberative robotic task planning," *Artificial Intelligence Review*, vol. 56, no. 9, pp. 9011–9049, 2023.

[17] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: Progress and challenges," *AI communications*, vol. 29, no. 1, pp. 151–162, 2016.

[18] B. Araki, K. Vodrahalli, T. Leech, C.-I. Vasile, M. Donahue, and D. Rus, "Learning to plan with logical automata," in *Robotics: Science and Systems*, 2019.

[19] C. I. Vasile, X. Li, and C. Belta, "Reactive sampling-based path planning with temporal logic specifications," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 1002–1028, 2020.

[20] Y. Zhao, Y. Li, L. Sentis, U. Topcu, and J. Liu, "Reactive task and motion planning for robust whole-body dynamic locomotion in constrained environments," *The International Journal of Robotics Research*, vol. 41, no. 8, pp. 812–847, 2022.

[21] S. Li and N. T. Dantam, "A sampling and learning framework to prove motion planning infeasibility," *The International Journal of Robotics Research*, vol. 42, no. 10, pp. 938–956, 2023.

[22] T. Pan, R. Shome, and L. E. Kavraki, "Task and motion planning for execution in the real," *IEEE Transactions on Robotics*, 2024.

[23] Y. Kantaros, M. Malencia, V. Kumar, and G. J. Pappas, "Reactive temporal logic planning for multiple robots in unknown environments," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 11 479–11 485.

[24] D. Sun, J. Chen, S. Mitra, and C. Fan, "Multi-agent motion planning from signal temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3451–3458, 2022.

[25] J. Fu, N. Atanasov, U. Topcu, and G. Pappas, "Optimal temporal logic planning in probabilistic semantic maps," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 3690–3697.

[26] Z. Ravichandran, V. Murali, M. Tzes, G. J. Pappas, and V. Kumar, "SPINE: Online semantic planning for missions with incomplete natural language specifications in unstructured environments," *arXiv preprint arXiv:2410.03035*, 2024.

[27] Y. Chen, J. Arkin, C. Dawson, Y. Zhang, N. Roy, and C. Fan, "Autotamp: Autoregressive task and motion planning with LLMs as translators and checkers," in *IEEE International Conference on Robotics and Automation*, 2024, pp. 6695–6702.

[28] W. Guo, Z. Kingston, and L. E. Kavraki, "CaStL: Constraints as specifications through LLM translation for long-horizon task and motion planning," *arXiv preprint arXiv:2410.22225*, 2024.

[29] D. Angluin, "Learning regular sets from queries and counterexamples," *Information and Computation*, vol. 75, no. 2, pp. 87–106, 1987.

[30] M. Vazquez-Chanlatte, K. Elmaaroufi, S. J. Witwicki, and S. A. Seshia, "L*LM: Learning automata from examples using natural language oracles," *arXiv preprint arXiv:2402.07051*, 2024.

[31] R. C. Carrasco and J. Oncina, "Learning deterministic regular grammars from stochastic samples in polynomial time," *RAIRO-Theoretical Informatics and Applications*, vol. 33, no. 1, pp. 1–19, 1999.

[32] D. Arrivault, D. Benielli, F. Denis, and R. Eyraud, "Scikit-SpLearn: a toolbox for the spectral learning of weighted automata compatible with scikit-learn," in *Conference francophone sur l'Apprentissage Aurtomatique*, 2017.

[33] C. Subakan, J. Traa, and P. Smaragdis, "Spectral learning of mixture of hidden Markov models," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[34] B. Balle and M. Mohri, "Learning weighted automata," in *International Conference on Algebraic Informatics*, 2015, pp. 1–21.

[35] M. Baert, S. Leroux, and P. Simoens, "Learning task specifications from demonstrations as probabilistic automata," *arXiv preprint arXiv:2409.07091*, 2024.

[36] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[37] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "LLM-planner: Few-shot grounded planning for embodied agents with large language models," in *IEEE/CVF International Conference on Computer Vision*, 2023, pp. 2998–3009.

[38] S. S. Kannan, V. L. Venkatesh, and B.-C. Min, "Smart-LLM: Smart multi-agent robot task planning using large language models," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2024, pp. 12 140–12 147.

[39] L. Guan, K. Valmeekam, S. Sreedharan, and S. Kambhampati, "Leveraging pre-trained large language models to construct and utilize world models for model-based task planning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 79 081–79 094, 2023.

[40] A. Ray, C. Bradley, L. Carlone, and N. Roy, "Task and motion planning in hierarchical 3d scene graphs," *arXiv preprint arXiv:2403.08094*, 2024.

[41] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[42] T. Michaud and M. Colange, "Reactive synthesis from LTL specification with spot," in *Workshop on Synthesis, International Conference on Computer-Aided Verification*, 2018.

[43] K. Grauman, A. Westbury, L. Torresani, K. Kitani, J. Malik, T. Afouras, K. Ashutosh, V. Baiyya, S. Bansal, B. Boote, *et al.*, "Ego-exo4d: Understanding skilled human activity from first-and third-person perspectives," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 19 383–19 400.

[44] M. Chevalier-Boisvert, B. Dai, M. Towers, R. Perez-Vicente, L. Willems, S. Lahlou, S. Pal, P. S. Castro, and J. Terry, "Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks," *Advances in Neural Information Processing Systems*, vol. 36, pp. 73 383–73 394, 2023.

[45] H. Mao, Y. Chen, M. Jaeger, *et al.*, "Learning deterministic probabilistic automata from a model checking perspective," *Machine Learning*, vol. 105, pp. 255–299, 2016.

[46] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[48] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," *arXiv preprint arXiv:2111.00396*, 2021.

[49] A. Dawid and Y. LeCun, "Introduction to latent variable energy-based models: a path toward autonomous machine intelligence," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2024, no. 10, p. 104011, 2024.