A Neurosymbolic Approach to Adaptive Feature Extraction in SLAM

Yasra Chandio[1], Momin A. Khan[1], Khotso Selialia[1], Luis Garcia[2], Joseph DeGol[3], and Fatima M. Anwar[1]

***Abstract*— Autonomous robots, autonomous vehicles, and humans wearing mixed-reality headsets require accurate and reliable tracking services for safety-critical applications in dynamically changing real-world environments. However, the existing tracking approaches, such as Simultaneous Localization and Mapping (SLAM), do not adapt well to environmental changes and boundary conditions despite extensive manual tuning. On the other hand, while deep learning-based approaches can better adapt to environmental changes, they typically demand substantial data for training and often lack flexibility in adapting to new domains. To solve this problem, we propose leveraging the neurosymbolic program synthesis approach to construct adaptable SLAM pipelines that integrate the domain knowledge from traditional SLAM approaches while leveraging data to learn complex relationships. While the approach can synthesize end-to-end SLAM pipelines, we focus on synthesizing the feature extraction module. We first devise a domain-specific language (DSL) that can encapsulate domain knowledge on the essential attributes for feature extraction and the real-world performance of various feature extractors. Our neurosymbolic architecture then undertakes adaptive feature extraction, optimizing parameters via learning while employing symbolic reasoning to select the most suitable feature extractor. Our evaluations demonstrate that our approach, neurosymbolic Feature EXtraction (`nFEX`), yields higher-quality features. It also reduces the pose error observed for the state-of-the-art baseline feature extractors ORB and SIFT by up to 90% and up to 66%, respectively, thereby enhancing the system's efficiency and adaptability to novel environments.**

I. Introduction

Accurate tracking is crucial to the wide-scale and practical deployment of autonomous cars, autonomous robots, and mixed-reality applications involving humans [1]. The different agents (e.g., cars, robots, humans) navigate complex and dynamic settings, from busy city streets to cluttered warehouses and ever-evolving homes. To ensure safe and effective operation, the agents require robust tracking mechanisms that can adapt to the constant changes they encounter.

Traditional Simultaneous Localization and Mapping (SLAM) often struggle in such dynamic scenarios. They rely on pre-defined physical models that may not generalize well to unseen situations or environments with significant lighting variations and scene transitions [2]. Recent advances in artificial intelligence (AI) driven approaches learn non-trivial relationships from data and perform better in dynamic environments and edge cases [3]. However, they are data-hungry and lack interpretability [4], hindering their deployment in safety-critical applications. Hybrid approaches have aimed to bridge the benefits of data-efficient, physics-based approaches and deep-learning-based approaches, e.g., by preprocessing the input to aid tracking [5] or placing guard rails around the tracking output to limit the impact of erroneous outcomes [6], [7]. While effective, the composition of these methods is often ad-hoc and tuned to specific domains or environments and not suited to adaptation [8].

In this work, we formalize these hybrid compositional approaches as neurosymbolic programs [9], [11], which aim to bridge the gap between data-driven learning approaches and rule-based symbolic reasoning. In particular, formulating the SLAM pipeline as a composition of modules–each potentially represented as a neurosymbolic program–enables neurosymbolic program synthesis, where we aim to synthesize tracking programs given a library of neural and symbolic components that fit a dataset and generalize to unseen inputs [9]. While neurosymbolic program synthesis can be employed at the module level or the entire tracking pipeline, we focus on the feature extraction module of the SLAM pipeline as a proof of concept. The feature extraction module detects and tracks feature maps over time. Significant prior work has been on developing feature extraction approaches, such as ORB [12] and SIFT [13]. These techniques are well-suited for certain environmental conditions for specific applications [14], [15]. Therefore, the feature extraction performance can be improved by dynamically selecting the most appropriate feature extractor and adapting its configuration parameters for the given environmental scenario. However, the search space across feature extractors, configurations, agent types, and scenarios is large, with limited performance data available for all combinations.

In this work, we consider finding the right feature extractor and its parameters a neurosymbolic program synthesis task. The *neural component* of our neurosymbolic feature extractor, `nFEX`, is a standard neural network that finds the optimal parameters for the various feature extractors we consider using a dataset of optimal configurations under different environmental conditions. We empirically generate this dataset by running exhaustive combinations of feature extractors and their parameters through SLAM pipelines. `nFEX`'s *symbolic component* captures the domain knowledge on the attributes of essential features and the impact of environmental conditions on the end-to-end performance of a SLAM pipeline as well as the feature quality metrics, such as texture, dissimilarity, and spatial density. Moreover, the symbolic representation can incorporate prior knowledge on the end-to-end performance of feature extractors; for example, ORB gives the lowest pose error under bright conditions with good textures. As with any neurosymbolic program

[1]Yasra Chandio, Momin A. Khan, Khotso Selialia, and Fatima M. Anwar are affiliated with the University of Massachusetts Amherst, USA.

[2]Luis Garcia is affiliated with the University of Utah, USA.

[3]Joseph DeGol is affiliated with the Steg AI, USA.

Correspondence: ychandio@umass.edu

synthesis task, we specify the *symbolic component* using the syntax of a domain-specific language (DSL) [16], and we leverage knowledge graphs to represent any symbolic expert knowledge. Given these components, we devise a learning algorithm that uses symbolic reasoning to adapt the outcomes of the neural learning component. The task for the algorithm is to discover the program's discrete architecture α (i.e., a feature extractor) and its real-valued parameters Θ (i.e., feature extractor's configurations). The task specification that directs this search includes a domain-specific quantitative fitness function derived from labeled data or expert knowledge. The algorithm aims to find a program that optimizes the loss under constraints.

The task of neurosymbolic program synthesis is not trivial and requires solving multiple challenges. First, there are no existing SLAM pipelines or feature extraction module DSLs. Therefore, we need to devise a DSL to represent various agent types (car, drone, human), their motion types (fast, slow), scene types (indoor, outdoor), light conditions (bright, dark), the set of feature extractors (ORB, SIFT), and their parameters (e.g., no. of features, density), details in §IV. Second, we must populate a knowledge graph summarizing how different feature extraction approaches perform under various conditions. However, as such data is limited in the existing literature [14], [15], we employed an experiment-driven approach to populate the knowledge graph (details in §III-A.1). We present our solution to those mentioned above and other challenges in their respective sections.

We make the following contributions in devising a neurosymbolic program synthesis approach to domain adaptive feature extraction in SLAM pipelines.

1) We develop a DSL for the feature extraction module of a SLAM pipeline that allows expressing the performance of a feature extractor across various scene characteristics. Since limited performance data is available on feature extractors, we leverage an experiment-driven approach to populate the knowledge graph required for the neurosymbolic program synthesis.
2) We devise a neural network-based learning approach for feature extractor synthesis that learns deep representations over the set of feature extractors and their parameters. We use a novel two-step fitness function (details in §IV) to direct the search toward an optimal feature extractor and its configuration parameters.
3) We extensively evaluate our proposed approach across three datasets, representing various agent and environmental characteristics and two underlying feature extractors, ORB and SIFT. We demonstrate that our approach dynamically adapts the feature extractor and its parameters. We outperform ORB and SIFT by 90% and 66%, using ATE as a metric (details in §V).

II. Background & Related Work

A. Feature Extraction in SLAM

Feature extraction acts as SLAM's eyes, the first module to encounter the raw environmental data. It encodes environmental elements for processing, determining how well it can detect and use landmarks to map and navigate surroundings, influencing the quality of the entire pipeline. This process involves operations such as scale-space representation, keypoint detection, transform invariance, and descriptor generation, enabling consistent landmark recognition across observations through techniques such as SIFT and ORB.

First, scale-space representation, utilizing image downsampling and filtering—Gaussian blur for SIFT and image pyramids [17] for ORB—creates a multi-resolution image pyramid, ensuring feature detection across all sizes in the visual field. Keypoint detection then identifies distinctive locations, such as corners, edges, or blobs, by targeting high-contrast areas using methods like the Harris Corner Detector [18], Difference of Gaussians [19] and Hessian matrix [20] to locate keypoints. Transformation invariance enhances this process by normalizing the region around each detected feature. SIFT, for example, determines orientation from local image gradients, rendering descriptors rotation invariant across views. On the other hand, ORB achieves invariance through intensity centroid-based methods. For each detected keypoint, a unique descriptor facilitates feature matching across images, with SIFT employing gradient histograms and a binary descriptor strategy for ORB [21].

B. Environmental Influences on Feature Extraction

Feature extraction is influenced by environmental changes (light, motion, reflective surfaces, textures), the specific agents navigating it (humans, cars, drones), and the scene's characteristics (indoors or outdoors). This causes several challenges. Lighting is one of several challenges. Too much light can hide image details, while too little makes features hard to detect and match. Motion can blur and displace critical features, complicating their tracking and matching across frames. Reflective surfaces can create false features, affecting map accuracy; well-textured scenes can confuse algorithms, while low-textured environments may offer few features to track, requiring careful selection of distinct features [22].

Furthermore, challenges of agent type and scene, whether navigating high-speed scenarios with cars, adapting to altitude and orientation changes in drones, or enhancing augmented reality experiences in complex indoor settings [23]. This customization extends to recognizing features like corners, edges, and textural patterns in indoor environments, tailored to the unique characteristics of man-made structures [24], while also being adaptable to handle the varying lighting, weather, and natural textures encountered outdoors. Both indoor and outdoor scenes require adjusting to dynamic environmental conditions and the specific demands of more complex environments, such as navigating through low visibility underwater or areas with repetitive manmade patterns. This highlights the need for SLAM systems to adapt to environmental changes and the inherent scene characteristics across different contexts.

Traditional methods struggle with these challenges, affecting feature quality and downstream applications (pose and maps). Adaptive techniques like thresholding and dynamic range adjustment [25], [26], optical flow, gyroscopic integration [27], polarization [28] and consistency checks [22],

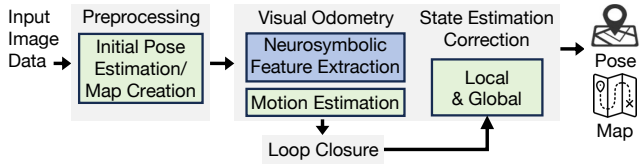

Fig. 1: *Our contribution (blue box) within SLAM pipeline.*

attempt to mitigate these issues but often fall short in rapidly changing conditions. However, these methods often fail in rapidly changing conditions primarily because they tend to be ad-hoc solutions or overly specialized to certain environments. Each technique is tailored to address particular issues, such as lighting variations or motion blur, without a holistic understanding of the environment's complexity.

C. Neurosymbolic Architectures

Recent studies [11], [29], [10] have highlighted the effectiveness of *neurosymbolic program synthesis* [30] in advancing the capabilities of machines to understand [31], [10], navigate [32], and interact with their environments [33], demonstrating a practical approach to enhancing feature extraction through the integration of learning and reasoning processes. Neurosymbolic architectures can address the main issues of usual SLAM feature extractors, which can not easily adjust to various environmental changes, affecting their accuracy and reliability. This approach uses the strength of neural networks to process complex data and symbolic systems to apply specific knowledge and rules [9], making it better suited for different conditions.

Integrating knowledge graphs, DSL, and grammar into SLAM presents a practical realization of neurosymbolic architecture. Knowledge graphs can organize unstructured environmental data and sensor interpretations into a structured format as a dynamic reference point. Using this graph, a DSL can provide a framework for expressing operations and configurations of feature extraction that align with the domain's specific requirements. To ensure coherence and semantic integrity, the grammar of a DSL can dictate the rules that structure the language, defining how symbols, keywords, and operators combine to form valid and meaningful expressions. This approach can fine-tune feature extraction parameters for challenges like low-light visibility and navigating reflective surfaces and select the best extractor for the situation. For instance, by symbolically analyzing motion and texture, it can learn to adjust motion compensation and feature quantity, steering towards extractors ideal for varied textures.

III. DSL Construction for Neurosymbolic SLAM

This section details nFEX approach, leveraging neurosymbolic programming to adapt feature extraction dynamically.

A. Neurosymbolic Program

In traditional SLAM, images are processed by a pre-selected feature extractor with fixed parameters, producing a feature vector fed to the remaining SLAM pipeline (odometry and other optimizations) for pose generation. To this end, our

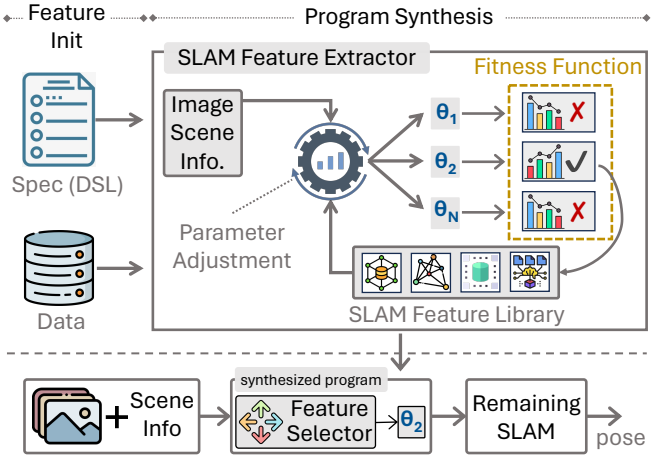

Fig. 2: *A high-level overview of our approach.*

proposed approach nFEX illustrated in Figure 2, synthesizes a neurosymbolic program designed to dynamically select and configure feature extractors based on real-time environmental inputs by replacing the traditionally manually-tuned feature extraction module with our *neurosymbolic feature extraction*, highlighted in blue color in Figure 1.

1) DSL Framework: This process begins with establishing a DSL framework, which outlines the structure for:
Feature Extractor Selection (α) identifies the optimal feature extractor architecture for the given scene conditions.
Parameter Configuration (Θ) determines the best parameter settings to enhance feature extractor performance.

It encapsulates domain insights into the feature extraction module with a knowledge graph that acts as a database of domain information that helps understand and navigate the module's operation and parameters, as illustrated in Figure 3; we generate this graph as one of our contributions. It encompasses nodes representing various feature extraction operations, such as scale space representation, keypoint detection, and descriptor. Edges within the graph delineate the relationships and compatibilities between these operations, informed by performance metrics, computational considerations, and adaptability to environmental factors. This graph enriches both our synthesis and the DSL's capacity to incorporate novel feature extraction methods in the future.

Next, *grammar*, outlined in Figure 5, provides a blueprint for creating the DSL program that can interpret and act upon input conditions (environmental context). Using the graph, the grammar parses the structure for input conditions (such as indoor/outdoor scene, agent types, lighting conditions, and so on), and each rule describes a part of the program for selecting the suitable α and their Θ. Our DSL's grammar enhances adaptability and streamlines each recalibration for new applications, making it straightforward for users to add new input conditions and parameters.

This DSL ensures the synthesized programs are logically coherent and actionable, allowing for automated parsing and execution of feature extraction. An example of one such program is illustrated in Figure 4. Once the DSL is established,

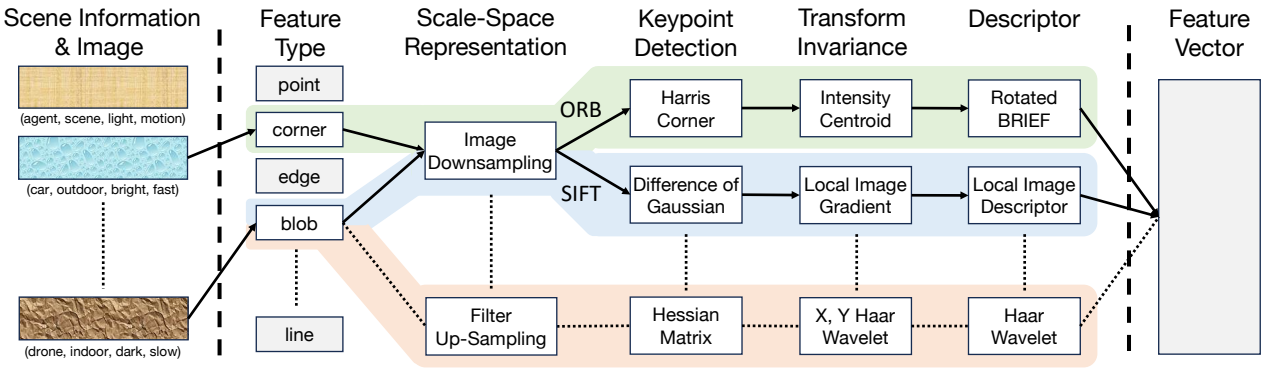

Fig. 3: *Illustration of feature extraction knowledge graph.*

we operationalize it through a two-phase process:

```
InputConditions {
    Scene: "Indoor" | "Outdoor";
    Agent: "Car" | "Human" | "Drone";
    LightType: "Bright" | "Dark";
    MotionType: "Fast" | "Slow";
    reflectiveSurface: "Yes" | "No";
    texture: "High" | "Low";
}
FeatureExtractionParameters {
    NF: Int = 1000;
    NL: Int = 4;
    SF: Float = 0.8;
    ST: Float = 6.0;
}
ParameterAdjustment {
    If LightType=="Bright" && MotionType=="Fast":
        If reflectiveSurface=="Yes" && texture=="High":
            NF: 338;
        ElseIf reflectiveSurface=="No" &&
        ↪  texture=="Low":
            NF: 1200;
        Else:
            NF: default;
    ElseIf LightType=="Dark" && MotionType=="Slow":
        NF: 800;
    Else:
        NF: default;
}
FeatureFeatureExtractorSelectionMetrics {
    Metrics:["texturedness", "stability", "motion",
    ↪  "dissimilarity", "spatialDensity",
    ↪  "distinctiveness"];
}
ComputeFeatureExtractorScore {
    InputImage: Image;
    Method: "SomeMethod";
}
FeatureExtractorSelection {
    If ComputeFeatureExtractorScore["SIFT"] >
    ↪  ComputeFeatureExtractorScore["ORB"]:
        Extractor: "SIFT";
    Else:
        Extractor: "ORB"; // default extractor
}
```

Fig. 4: Example DSL program

This DSL ensures the synthesized programs are logically coherent and actionable, allowing for automated parsing and execution of feature extraction. An example of one such program is illustrated in Figure 4. Once the DSL is established, we operationalize it through a two-phase process:

2) Program Synthesis (Training): We first optimize feature extractor parameters (Θ) using neural networks (details in § IV) to generalize the extractor's performance across different contexts. To handle the inherent uncertainties and partial unstructured information in real-world scenarios, we employ smooth interpretation [34] techniques, allowing for a nuanced adaptation. Then, with this optimized Θ, the program evaluates to make informed decisions about the most suitable α and its configuration for the given context. This selection process is akin to neural architecture search [35], where various architectures (feature extractors in this context) are evaluated for contextual performance. Finally, we employ the feedback mechanism integral to this phase, allowing continuous program refinement based on downstream performance (e.g., pose). This iterative process ensures the program remains optimally configured to adapt to new challenges and environments.

```
<DSLProgram> ::= <InputConditions><
    ↪ FeatureExtractionParameters><ParameterAdjustment>
    ↪ <FeatureExtractorSelection>

<InputConditions> ::= <Condition>*
<Condition> ::= <ConditionKey>":"<ConditionValue> ";"

<ConditionKey> ::= "Scene"| .. |"Texture"
<ConditionValue> ::= "Indoor"|"Outdoor"| .. |"High"|"Low"

<FeatureExtractionParameters> ::= <Parameter>*
<Parameter> ::= <ParamKey>":"<ParamValue> ";"
<ParamKey> ::= "NF" | "NL" | "SF" | "ST"
<ParamValue> ::= <Number>

<ParameterAdjustment> ::= "If " <Expression> "{" <
    ↪ ConditionKey>":"<ConditionValue>";" "}"
<Expression>::=<ParamKey>"=="<Number>

<FeatureExtractorSelection>::= Extractor":"<Extractor>";"
<Extractor> ::= "SIFT" | "ORB"
```

Fig. 5: Grammar to parse the DSL for nFEX.

3) Program Use (Inference): During inference, our synthesized program dynamically applies optimized Θ) and α in real-time scenarios. It dynamically adjusts to changing environments, analyzing scene data for accurate feature extraction and integration into the SLAM pipeline. This phase

emphasizes real-time adaptation and continuous learning from new data, enabling ongoing system performance refinement and boosting adaptability across diverse scenarios.

IV. Implementation

A. Fitness Function for DSL formulation

We implement a DSL program through a two-step fitness function to materialize our approach.

1) Parameter Tuning (Θ): Given a set of feature extractors $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_y\}$, initial parameters $\Theta = \{\vartheta_1, \vartheta_2, \ldots, \vartheta_n\}$ for each feature extractor and a set of *symbolic* environmental conditions $E = \{e_1, e_2, \ldots, e_c\}$ the optimization process is defined by a transformation function $g(\vartheta, E)$, which *dynamically adjusts* Θ based on E, resulting in an optimized parameter set $\Theta' = \{\vartheta'_1, \vartheta'_2, \ldots, \vartheta'_n\}$ with w_k, the weight for each parameter k:

$$\Theta' = g(\Theta, E), \quad w_k = \prod_{j=1}^{c} f_{kj}(e_j)$$

$$\vartheta'_k = w_k \cdot \vartheta_k = \left(\prod_{j=1}^{c} f_{kj}(e_j) \right) \cdot \vartheta_k \quad \text{for each } k \in \{1, 2, \ldots n\}$$

The term $\prod_{j=1}^{c} f_{kj}(e_j)$ is the cumulative product of adjustments from all the environmental conditions for a given parameter k. Finally, the fitness of the optimized parameters ϑ' in the context of environmental conditions E is then evaluated by the fitness function F:

$$F(\Theta'|E) = f(\Theta'|E)$$

This formulation demonstrates how scene conditions directly influence parameter optimization, allowing for adaptable feature extraction based on the environment.

2) Feature Extractor Selection (α): Once we obtain Θ' for each feature extractor α_i, given set of *symbolic* metrics $M = m_1, m_2 \ldots m_p$ with dynamically adjusted weights $\rho_x = h_x(E)$ for each $x \in \{1, 2, \ldots, p\}$. $h_x(E)$ is a function that computes the weight of the x^{th} metric based on the environmental conditions E. We assess the quality of detected features for each feature extractor α_i with its parameters optimized as Θ'_i:

$$F_{metrics}(\alpha_i|\Theta'_i, E) = \sum_{x=1}^{p} \rho_x \cdot m_x(\alpha'_i) = \sum_{x=1}^{p} h_x(E) \cdot m_x(\alpha'_i)$$

The final selection of the feature extractor α^* that maximizes $F_{metrics}$:

$$\alpha^* = \underset{\alpha_i \in \{\alpha_1, \alpha_2, \ldots, \alpha_y\}}{\arg\max} F_{metrics}(\alpha_i|\Theta'_i, E)$$

This ensures that the feature extractor's selection and performance evaluation are adapted to the environmental context.

B. Experimental Setup

1) Feature Parameters (Θ): Our approach prioritizes optimizing parameters that influence feature extraction operations and are responsive to our defined environmental inputs. To this end, we choose the following parameters:

Number of Features **(NF)** affects the system's ability to recognize and track environmental landmarks. Optimizing NF is crucial for balancing detail capture with computational load, ensuring enough features are detected to represent the scene accurately without overwhelming the processing capacity, particularly in complex or sparse environments.

Scale Factor **(SF)** determines the interval between successive scales in the image pyramid, influencing the system's capacity for scale-invariant feature detection. A properly tuned SF allows the feature extractor to consistently identify features regardless of their size in the image, enabling the system to remain robust to changes in the size or distance of objects affected by lighting conditions and motion.

Number of Pyramid Levels **(NL)** adjusts for features across object sizes. It expands the system's capability to discern features of various sizes across the entire scene, directly impacting the system's versatility in handling scenes with diverse spatial characteristics.

Keypoint Selectivity Thresholds **(ST)**, ensure distinct, reliable keypoints for matching across images, crucial for dealing with reflective surfaces or inconsistent textures by promoting uniform feature distribution.

2) Feature Extractors (α): We integrated our neurosymbolic feature extraction module into ORBSLAM3 [36] due to its advanced adaptability and modular design, enabling effective evaluation of our `nFEX` approach within its framework. By selecting both ORB for its computational efficiency in corner detection and SIFT for its precision in blob detection, we aim to assess the distinct advantages each feature extractor brings to the SLAM. This approach allows for a comprehensive comparison of corner-based and blob-based feature extraction methods, leveraging our fitness functions to optimize and evaluate their performance.

*3) Feature quality metrics (*M*):* Our choice of M focuses on metrics intrinsic to feature extraction, allowing our neurosymbolic optimization to influence them without relying on downstream SLAM data.

Texturedness ($m_1 = \sqrt{\sum (I(a,b) - \bar{I})^2}$) integrates texture variance within the image with $I(a, b)$ being pixel intensity and $\bar{I}$ the average intensity.

Dissimilarity ($m_2 = \sum |I_{feature} - I_{surrounding}|$) measures feature uniqueness against its surroundings.

Motion ($m_3 = \|\mathbf{a}_{t+1} - \mathbf{a}_t\|$) quantifies feature movement between frames.

Stability ($m_4 = \frac{\text{Number of stable features}}{\text{Total number of features}}$) evaluates detection consistency over time.

Spatial density ($m_5 = \frac{\text{Number of Features}}{\text{Area of Region}}$) examines evenness of feature distribution.

Distinctiveness ($m_6 = \frac{1}{\sum \text{similarity(feature, other features)}}$) gauges how features stand out from each other.

Repeatability ($m_7 = \frac{\text{Number of Re-detected Features}}{\text{Total Number of Features}}$) measures consistent feature detection.

4) Datasets: We use three datasets to evaluate adaptability across various agents, scenes, and motion types: KITTI [37], for urban outdoor scenes with fast-moving cars; EuRoC [38], focusing on diverse indoor environments navigated by drones with dynamic motions; and HoloSet [39], covering both indoor and outdoor settings from a human perspective, capturing natural human motions.

TABLE I: Performance comparison of different feature extractors using Mean ATE in meters (averaged over 10 runs).

		EuRoC		KITTI		HoloSet	
		MH01	MH05	KITTI-1	KITTI-6	Campus-Center-1	Suburb-Jog-2
ORB	Default	0.855	0.952	2.955	1.173	11.789	11.604
	Dynamic	0.792	0.815	**0.565**	0.126	5.903	5.845
SIFT	Default	0.860	1.038	*fail*	*fail*	13.789	12.825
	Dynamic	0.859	0.882	6.426	5.875	7.049	6.984
nFEX		**0.704**	**0.761**	**0.565**	**0.115**	**4.729**	**5.800**

Our approach requires scene attributes to adapt dynamically to diverse environments. Because they significantly impact feature extraction, we selected specific scene condition labels, such as indoor/outdoor settings, agent types, lighting conditions, motion speeds, reflective surfaces, and texture levels. We derived these labels through methods proposed in [40] and corroborated with scene information from [41], [38], and [39]. However, we acknowledge that creating exhaustive verification mechanisms for dataset characterization against real-world scenarios is an open problem and beyond the scope of this paper.

5) Fitness Function Training: We use a two-step neural network to optimize feature extraction parameters $F(\Theta'|E)$ and feature extractor selection α^*. $F(\Theta'|E)$ training uses a Multi-Layer Perceptron (MLP) fully connected layers ($\times 3$) and ReLU activations ($\times 2$), minimizing the loss as:

$$\mathcal{L}(\Theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i(\Theta))^2$$

where N is the batch size, y_i the true label, and $\hat{y}_i(\Theta)$ the predicted value. Next, α^* training uses a hybrid neural network combining Convolutional Neural Network (CNN) layers for image analysis and fully connected layers ($\times 3$) for numerical data, refining feature quality by minimizing the cross-entropy loss, which quantifies the discrepancy between predicted probabilities $\hat{l}_i(\alpha^*)$ and true labels l_i as:

$$\mathcal{L}(\alpha^*) = -\frac{1}{N}\sum_{i=1}^{N}\left[l_i \log \hat{l}_i(\alpha^*) + (1-l_i)\log(1-\hat{l}_i(\alpha^*))\right]$$

Feature scaling is applied using a standard scaler to ensure features have a mean of zero and a standard deviation of one, making them more suitable for neural network training.

V. Evaluation

We evaluate the integration of nFEX with ORBSLAM3 by comparing performance using the Absolute Trajectory Error (ATE). It calculates the deviation between aligned estimated T_{est} and ground truth trajectories T_{gt} as: $ATE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|T_{gt}(i) - T_{est}(i)\|^2}$. Here N represents the total number of trajectory points being compared between the T_{est} and T_{gt}. Though precise trajectories do not guarantee flawless maps, they are widely used to gauge SLAM performance. We use ORB and SIFT with default (parameters from OpenCV) and dynamically optimized parameters (Θ) as baselines, and we trained our models on sequences EuroC-MH_02, KITTI-00, and Holoset-Suburb-seq-1. Lastly, we ran all our tests on NVIDIA GeForce RTX 2070 GPU, with a batch size of 8, Adam optimizer, and a learning rate of $1e^{-4}$.

TABLE II: Generalization performance of nFEX when trained on the first (or part of a) sequence and tested on the second sequence in the column heading.

	MH01–MH01	MH01–MH05	MH01 – KITTI-6
nFEX	0.792 m	1.329 m	2.476 m

A. End-to-End Performance

Table I showcases the Mean ATE for ORB and SIFT feature extractors with default and dynamically optimized parameters alongside our nFEX approach across sequences from all three datasets. We can see that dynamic optimization significantly enhances ORB's performance, notably in the KITTI sequences, where it drastically lowers Mean ATE, underscoring the importance of parameter tuning. However, SIFT fails in its default mode for KITTI and shows limited improvement even when dynamically optimized.

In challenging indoor scenarios of EuRoC and mixed environments of HoloSet, nFEX achieves the lowest Mean ATE, highlighting its effectiveness in complex settings. Even after optimization, SIFT's failure in KITTI's default configuration and its lagging performance also highlights nFEX's edge in achieving a balance between robust feature extraction and computational efficiency. Overall, nFEX shows its potential to significantly enhance SLAM by adapting to the operational environment, choosing the best feature extractor, and optimizing parameters dynamically.

B. Generalization

Table II highlights the generalization capability of nFEX. In the first case, MH01 – MH01, the training on the first 70% of the sequence and testing on the remaining 30% of the sequence yielded a mean ATE of 0.792m, demonstrating nFEX's effectiveness within familiar environments. When tested on MH05 (difficult sequence) after training on MH01 (easy sequence), the ATE increased to 1.329m, indicating a reasonable generalization to a different sequence within the same dataset despite new environmental and motion dynamics. The more significant jump to 2.476m when transitioning from training on MH01 to testing on KITTI-6 reflects the challenge of adapting to entirely different environmental conditions, such as outdoor versus indoor settings and drone versus car motion dynamics. These results underscore nFEX's potential for cross-environment generalization.

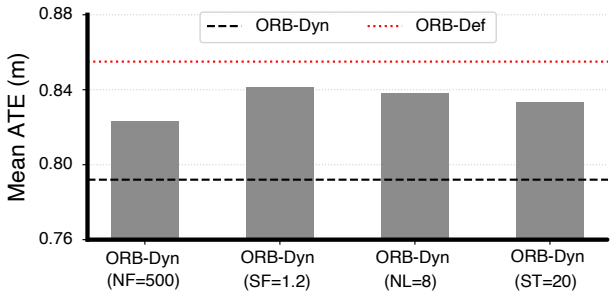

Fig. 6: Mean ATE in meters for ORB-Dyn where one parameter is fixed at a time.

TABLE III: Average number of feature matches for different feature extractors for MH01 sequence.

	ORB-Def	ORB-Dyn	SIFT-Def	SIFT-Dyn	nFEX
# of features	500	418	no limit	518	529
# of matches	389	602	1101	328	772

C. Ablation Study

1) Impact of Parameter Optimization: In Figure 6, we examine the effect of individual parameters—NF, SF, NL, ST—on the ATE for the ORB feature extractor. We fixed one parameter and kept the other dynamic. ORB with all dynamic or all default parameters are highlighted by black and red lines, respectively. This comparison shows the potential of dynamic parameter configurations, particularly in complex and varied environments. Further analysis shows nFEX and dynamically optimized ORB (ORB-Dyn) significantly increase feature matches over default settings (Table III). nFEX notably outperforms default ORB (ORB-Def) and SIFT with dynamic parameters (SIFT-Dyn), showcasing its ability to optimize feature extraction for superior match quality adaptively.

2) Impact of Feature Extractor Selection: Table IV illustrates nFEX's feature extractor selection on the MH01 sequence, which dynamically switches between ORB and SIFT extractors. The selection process results in a diverse distribution: ORB-Dyn is selected for 1366 out of 3638 frames, demonstrating its preference under specific conditions, followed by SIFT-Dyn with 1124 frames. In contrast, ORB-Def and SIFT-Def (SIFT with default parameters) are chosen less frequently, indicating that dynamic parameter optimization generally outperforms default settings.

D. Computation

In Table V, we show the computational efficiency of nFEX averaged over 10 runs. Model loading takes 1800 milliseconds (ms), a one-time overhead. Parameter and feature extractor selection are executed swiftly, ensuring rapid adaptability. At 112 ms, frame processing remains within acceptable bounds for real-time application, positioning nFEX as a viable solution for enhancing SLAM performance with negligible impact on processing speed.

VI. Limitations and Future Work

Data efficiency. Demonstrating significant data efficiency, nFEX aligns with prior work [42], suggesting a minimal data subset can yield substantial system efficiency. Enhancing this efficiency further will make nFEX adaptable to diverse applications, particularly where data collection is challenging.

Interpretability. nFEX's dynamic extractor selection for varying scenes enhances model transparency to a certain degree. By analyzing time series data, we can illustrate how and why certain extractors are chosen, offering insights based on logical reasoning derived from the context. This is crucial for critical applications where understanding the model's decision-making process is as important as its performance. Moreover, incorporating physical models to offer performance guarantees akin to control systems will enhance nFEX's reliability and predictability in safety-critical applications such as autonomous driving.

Improved cross-environment generalization. While promising within similar datasets, nFEX faces some challenges across different environments, necessitating dataset-specific tuning for optimal performance. Addressing this will further our goal of universal SLAM adaptability.

Resource Optimization Tailoring nFEX to operate within the constraints of specific resources, as informed by the agent type (e.g., computational power available on a drone versus a car), is an area for improvement. Optimizing the program to utilize available resources efficiently can enhance deployment flexibility and operational efficiency, ensuring nFEX is effective across a wide range of use cases. nFEX's neurosymbolic architecture is well-suited for platform-aware neurosymbolic architecture search approaches, e.g., [43].

This work is the first step toward leveraging neurosymbolic programming for domain-adaptive SLAM. It guides the community toward tackling the outlined challenges and exploring open problems. By continuing to refine and build upon these concepts, we aim to steer future research toward developing more adaptable, efficient, and explainable SLAM.

TABLE IV: Selection of different feature extractors by nFEX for different frames within MH01 sequence.

	ORB-Def	ORB-Dyn	SIFT-Def	SIFT-Dyn
# of frames	613 / 3638	1366 / 3638	535 / 3638	1124 / 3638

TABLE V: Timing for tasks in nFEX

	Parameter Selection	Frame Processing	Extractor Selection
Time	5.49 µs	112 ms	4.37 µs

VII. Conclusion

Accurate and adaptive tracking remains a key challenge for achieving agents' safe and robust operation in dynamic environments. Our approach leverages the strengths of both symbolic reasoning and data-driven learning to dynamically select and configure the most appropriate feature extractor based on the encountered environment. We demonstrated how nFEX's dynamicity significantly improves performance across three representative benchmark datasets in different domains compared to traditional approaches. While this work focuses on adaptive SLAM feature extraction, the modular nature of the SLAM pipeline paves the way for extending neurosymbolic program synthesis to other modules as well.

ACKNOWLEDGMENTS

The research reported in this paper was sponsored in part by the National Science Foundation (NSF) under awards 2435642 and 2237485. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies.

REFERENCES

[1] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007.

[2] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, 2016.

[3] E. Jeong, J. Lee, and P. Kim, "A comparison of deep learning-based monocular visual odometry algorithms," in *Asia-Pacific International Symposium on Aerospace Technology*, pp. 923–934, Springer, 2021.

[4] Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang, "A survey on neural network interpretability," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 5, pp. 726–742, 2021.

[5] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers, "D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[6] C. Chen, C. X. Lu, B. Wang, N. Trigoni, and A. Markham, "Dynanet: Neural Kalman dynamical model for motion estimation and prediction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 12, pp. 5479–5491, 2021.

[7] Z. Xing, X. Zhu, and D. Dong, "DE-SLAM: Slam for highly dynamic environment," *Journal of Field Robotics*, vol. 39, pp. 528–542, 2022.

[8] C. Chen, B. Wang, C. X. Lu, N. Trigoni, and A. Markham, "A survey on deep learning for localization and mapping: Towards the age of spatial machine intelligence," *arXiv preprint arXiv:2006.12567*, 2020.

[9] S. Chaudhuri, K. Ellis, O. Polozov, R. Singh, A. Solar-Lezama, Y. Yue, *et al.*, "Neurosymbolic programming," *Foundations and Trends® in Programming Languages*, vol. 7, no. 3, pp. 158–243, 2021.

[10] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, "The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision," *arXiv preprint arXiv:1904.12584*, 2019.

[11] D. Ritchie, P. Guerrero, R. K. Jones, N. J. Mitra, A. Schulz, K. D. Willis, and J. Wu, "Neurosymbolic models for computer graphics," in *Computer Graphics Forum*, vol. 42, pp. 545–568, Wiley, 2023.

[12] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An Efficient Alternative to SIFT or SURF," in *International Conference on Computer Vision*, 2011.

[13] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[14] E. Karami, S. Prasad, and M. S. Shehata, "Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images," *CoRR*, vol. abs/1710.02726, 2017.

[15] M. Bansal, M. Kumar, and M. Kumar, "2D Object Recognition: A Comparative Analysis of SIFT, SURF and ORB Feature Descriptors," *Multimedia Tools and Applications*, vol. 80, pp. 18839–18857, 2021.

[16] M. Mernik, J. Heering, and A. M. Sloane, "When and how to develop domain-specific languages," *ACM Computing Surveys*, vol. 37, no. 4, pp. 316–344, 2005.

[17] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, "Pyramid methods in image processing," *RCA Engineer*, vol. 29, no. 6, pp. 33–41, 1984.

[18] K. G. Derpanis, "The harris corner detector," *York University*, vol. 2, pp. 1–2, 2004.

[19] B. Blair and C. Murphy, "Difference of Gaussian scale-space pyramids for sift feature detection," *Complex Digital Systems Design, Final report*, 2007.

[20] A. Sasson, F. Viloria, and F. Aboytes, "Optimal load flow solution using the hessian matrix," *IEEE Transactions on Power Apparatus and Systems*, no. 1, pp. 31–41, 1973.

[21] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *11th European Conference on Computer Vision*, pp. 778–792, Springer, 2010.

[22] L. Yan, X. Hu, L. Zhao, Y. Chen, P. Wei, and H. Xie, "DGS-SLAM: A fast and robust RGBD slam in dynamic environments combined by geometric and semantic information," *Remote Sensing*, vol. 14, no. 3, p. 795, 2022.

[23] M. Billinghurst, A. Clark, G. Lee, *et al.*, "A survey of augmented reality," *Foundations and Trends® in Human–Computer Interaction*, vol. 8, no. 2-3, pp. 73–272, 2015.

[24] S.-Y. An, J.-G. Kang, L.-K. Lee, and S.-Y. Oh, "Slam with salient line feature extraction in indoor environments," in *11th International Conference on Control Automation Robotics & Vision*, 2010.

[25] K.-F. Yang, H. Li, H. Kuang, C.-Y. Li, and Y.-J. Li, "An adaptive method for image dynamic range adjustment," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 3, 2018.

[26] F. Drago, K. Myszkowski, T. Annen, and N. Chiba, "Adaptive logarithmic mapping for displaying high contrast scenes," in *Computer Graphics Forum*, vol. 22, pp. 419–426, Wiley, 2003.

[27] M. Hwangbo, J.-S. Kim, and T. Kanade, "Inertial-aided klt feature tracking for a moving camera," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[28] Z. Zhu, P. Xiang, and F. Zhang, "Polarization-based method of highlight removal of high-reflectivity surface," *Optik*, vol. 221, 2020.

[29] E. Marconato, S. Teso, A. Vergari, and A. Passerini, "Not all neuro-symbolic concepts are created equal: Analysis and mitigation of reasoning shortcuts," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[30] E. Parisotto, A.-r. Mohamed, R. Singh, L. Li, D. Zhou, and P. Kohli, "Neuro-symbolic program synthesis," *arXiv preprint arXiv:1611.01855*, 2016.

[31] S. Kelly, D. S. Park, X. Song, M. McIntire, P. Nashikkar, R. Guha, W. Banzhaf, K. Deb, V. N. Boddeti, J. Tan, *et al.*, "Discovering adaptable symbolic algorithms from scratch," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3889–3896, IEEE, 2023.

[32] A. Siyaev and G.-S. Jo, "Neuro-symbolic speech understanding in aircraft maintenance metaverse," *IEEE Access*, vol. 9, pp. 154484–154499, 2021.

[33] T. Chen, Q. Wang, Z. Dong, L. Shen, and X. Peng, "Enhancing robot program synthesis through environmental context," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[34] S. Chaudhuri and A. Solar-Lezama, "Smooth interpretation," in *Proceedings of the 31st ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2010.

[35] T. Elsken, J. H. Metzen, and F. Hutter, "Neural Architecture Search: A Survey," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1997–2017, 2019.

[36] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An Accurate Oen-source Library for Visual, Visual-Inertial and Multi-map SLAM," *IEEE Transactions on Robotics*, vol. 37, no. 6, 2021.

[37] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision Meets Robotics: The Kitti Dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, 2013.

[38] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EUROC Micro Aerial Vehicle Datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, 2016.

[39] Y. Chandio, N. Bashir, and F. M. Anwar, "Holoset-a dataset for visual-inertial pose estimation in extended reality: Dataset," in *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, 2022.

[40] I. Ali and H. Zhang, "Are we ready for robust and resilient slam? A framework for quantitative characterization of slam datasets," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022.

[41] S. Guo, Z. Rong, S. Wang, and Y. Wu, "A lidar slam with PCA-based feature extraction and two-stage matching," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, 2022.

[42] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. Tenenbaum, "Neural-symbolic vqa: Disentangling reasoning from vision and language understanding," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[43] S. S. Saha, S. S. Sandha, M. Aggarwal, B. Wang, L. Han, J. d. G. Briseno, and M. Srivastava, "Tinyns: Platform-aware neurosymbolic auto tiny machine learning," *ACM Transactions on Embedded Computing Systems*, 2023.