Master 1 Advanced Robotics - Research Project

Learning Motion Patterns via IMU-Based Odometry with Neural Networks

May 15, 2024

Research presented by
Raphael Blanchard

to the Robotics Department of
Ecole Centrale de Nantes

Supervised by:
Sebastian Scherer, Shibo Zhao (Carnegie Mellon University)
Elwan Héry (Ecole Centrale de Nantes)

***Abstract*—We present a learning-based approach for odometry using solely Inertial Measurement Unit (IMU) data. The primary goal of our research is to enable a neural network to learn and capture the inherent motion patterns of an agent, thereby facilitating the development of higher intelligence navigation systems. By integrating components such as Periodic AutoEncoders (PAE) [1] and Stationary Wavelet Transforms (SWT) [2], our method extracts meaningful features from IMU data. This dual-focused approach not only enhances the accuracy and robustness of the odometry system but also allows for the extraction of valuable motion pattern information.**

I. Introduction

Odometry is a critical component of autonomous navigation, enabling robots to track their position and orientation over time. Traditional Visual Odometry (VO) [3], Visual-Inertial Odometry (VIO) [4] and Lidar-Inertial Odometry (LIO) [5] systems rely heavily on perception sensors such as cameras and lidars. However, these systems face significant challenges in environments where perception sensors struggle, such as low-light or featureless terrains. A notable example is NASA's Ingenuity helicopter [6] on Mars, which experienced landing issues on it's 72nd flight due to lack of visual features. This highlights the need for robust odometry solutions that can operate effectively regardless of the environment.

Inertial Measurement Units (IMU) are a type of sensor that provide motion-related data unaffected by environmental conditions. This makes them particularly valuable for odometry in challenging scenarios. However, using IMU data alone for odometry is not possible when dealing with complex motion and long trajectories due to high levels of noise and drift over time. Despite these challenges, recent advancements in deep neural networks (DNNs) offer new opportunities to enhance IMU-based odometry.

Our goal in this project was not only to improve the results of previous work on learning-based odometry, but also to enable our network to learn the robot's motion patterns. To achieve this, we used the work of Starke *et al.* [1] on Periodic AutoEncoders (PAE) to extract periodic features from the agent. We also implemented the Stationary Wavelet Transform (SWT), which allowed us to obtain information from both the low and high frequency components in the Time-Frequency domain.

Acquiring general information about the environment to enhance intelligent navigation systems is an active area of research. **Neural-Fly** [7], for instance, rapidly learned an effective representation of aerodynamics in different wind conditions, which then helped to control an Unmanned Aerial Vehicle (UAV) in strong winds. 3D scene graphs, as demonstrated in Hydra [8], enable robots to understand and categorize different elements of their environment. Similarly, our approach aims to provide robots with a more comprehensive understanding of their motion, moving towards the goal of achieving higher levels of robotic intelligence. To summarize, our contributions are:

- Proposing a network architecture that incorporates new components, based on previous work and achieving better results for odometry.

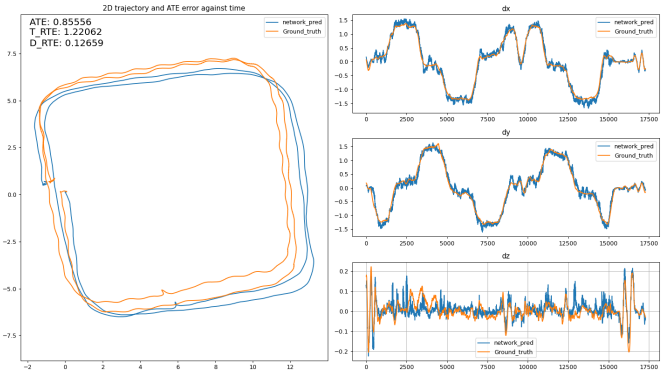


Fig. 1: Result on unseen data. Left plot is the 2D ground truth (orange) and predicted (blue) trajectories, right plots are the ground truth (orange) and predicted (blue) translations for the x, y and z axis.

- Enabling our network to learn and capture the motion patterns of the robot, providing valuable insights into the robot's dynamics.

II. Related Work

We primarily used RNIN-VIO [9], a robust odometry system using handheld data collected via a mobile phone. This framework utilizes a deep learning-based inertial network (RNIN) that relies on IMU measurements. They fuse the RNIN network with visual-inertial (VI) measurements within an Extended Kalman Filter (EKF) framework, enhancing pose prediction by combining diverse sources of data.

In our work, we focus specifically on the RNIN part. We use the base of the RNIN network (Resnet-LSTM) to design our architecture and compare our results with this network.

III. System Overview

A. System

Our proposed architecture is illustrated in Fig. 2. The inputs to the network include raw IMU data, the output of a forward pass in a pretrained Periodic AutoEncoder (PAE), and the outputs of the Stationary Wavelet Transform (SWT) for both linear acceleration and angular velocities. The network then predicts the global translation/displacement of the robot for the x, y, and z coordinates. We provide detailed explanations of the PAE and SWT, along with the reasons for their inclusion in our architecture, in Sections IV and V, respectively.

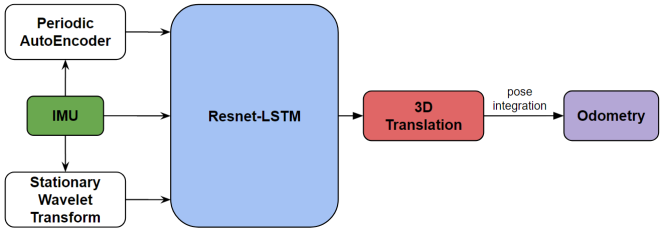


Fig. 2: System architecture

B. Network

Similarly to RNIN-VIO, our network consists of a 1D version of a ResNet18 to extract features from the motion, a LSTM and fully-connected layers (see Fig. 3). We process our data using windows of 1 second's worth of data at 200Hz, and use sequences of 2 windows as inputs to our network.

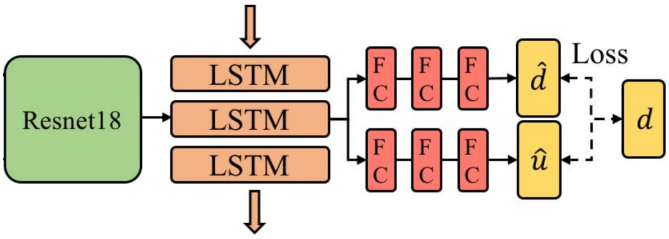


Fig. 3: Network architecture

The inputs to the Resnet-LSTM are as follows:

- Raw IMU data, 6 channels (3 for linear acceleration and 3 for angular velocities).
- The output of a pretrained PAE that was given angular velocities, 5 channels.
- Information from the Stationary Wavelet Transform (SWT), including 6 channels for angular velocities and 6 channels for linear accelerations.

C. Loss functions

To ensure the network focuses on both local accuracy and long-term global accuracy, we used three loss functions that we define in equation (1), (2) and (3).

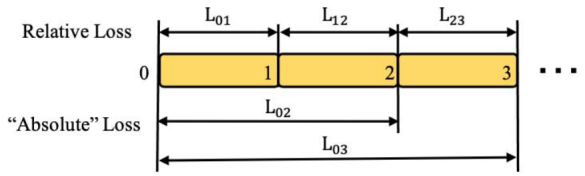


Fig. 4: Relative and Absolute losses

Relative Loss: To allow the network to learn the movement of a single window and improve the measurement accuracy of the direct output of the network, we add a relative loss (RL) function to the single window. The Mean Square Error (MSE) is chosen to optimize the loss. RL loss is defined as:

$$\mathcal{L}_{RL}^{MSE}(\mathbf{d}, \hat{\mathbf{d}}) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{M}\sum_{j=m}^{m+M}\left(d_j - \hat{d}_j\right)\right) \quad (1)$$

where $\hat{d}_j$ is the j-th window 3D translation output of the network and d_j is the corresponding ground truth. n is the batch size during training, and m is the start window of the sequence. M is the number of LSTM windows.

Absolute Loss: In addition to the accuracy of a single window, we are more concerned about the cumulative error over a longer period of time. To allow the network to learn the long sequence relationship of the motion, thereby reducing the long-term cumulative error, we designed the absolute loss function:

$$\mathcal{L}_{ALL}^{MSE}(\mathbf{d}, \hat{\mathbf{d}}) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{M}\sum_{j=m}^{m+L}\left(d_{m+j} - \hat{d}_{m+j}\right)\right) \quad (2)$$

Z-MSE: We've also used the MSE for the z axis to adapt the network to motions along the z axis. We've seen that without this loss the network lacked accuracy when predicting motions along the z axis. This loss is defined as:

$$\mathcal{L}_z(\mathbf{d}, \hat{\mathbf{d}}) = c * \frac{1}{n}\sum_{j=1}^{n}\left(dz_j - \hat{dz}_j\right)^2 \quad (3)$$

where dz_j is the Z translation of the j-th window, $\hat{dz}_j$ it's corresponding ground truth and c a coefficient to alter the weight of this function into our final loss as defined:

Final Loss: The final loss function is the sum of the relative loss, absolute loss, and z-axis MSE loss, and is defined as:

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{RL}^{MSE} + \mathcal{L}_{ALL}^{MSE} + \mathcal{L}_z \quad (4)$$

D. Implementation

We use PyTorch as the implementation of our model, and train it using the Adam optimizer with an initial learning rate of 0.0001. We use a subset of data provided by RNIN-VIO collected on a mobile phone, for a total of 4.5 hours of training data. It took 3 epochs for our network to reach a plateau, which took approximately 1h30 on an NVIDIA 3070ti.

IV. Periodic AutoEncoders

Periodic AutoEncoders (PAE) [1] employ a learning-based approach to effectively learn and represent periodic features in data through sine waves. In our case, the PAE learns the parameters of 5 sine waves for each sequence of data using the Fast Fourier Transform (FFT). The PAE then uses the sine waves to reconstruct the original signal and then evaluates the reconstruction loss with the Mean Squared Error to ensure that the learned parameters accurately capture the periodic characteristics of the data.

We chose to work with 2-second sequences because, upon computing the FFT on different durations of data, we observed that using 2 seconds of data provided the best reconstruction results when using the dominant frequency (see Fig. 5). This suggests that the data from angular velocities frequently exhibits periodic behavior within 2-second windows.

The PAE network was pretrained on a subset of trajectories from our training data. Once pretrained, we pass the raw angular velocities through the PAE and use the outputted sine waves for each window as additional inputs to the ResNet-LSTM.

We decided to use PAE because it allows for the extraction of meaningful information about the motion patterns of the agent. As demonstrated in their paper, the phase representation

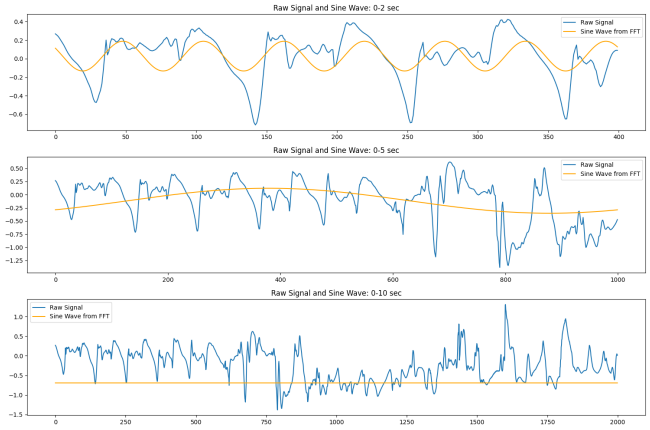


Fig. 5: Sine wave reconstruction from FFT (dominant frequency) on z-axis angular velocity obtained from a legged robot walking on rough terrain, using 2, 5 and 10-second data windows.

of periodic data captured by PAE provides a more informative and compact representation of the motion patterns compared to a convolution representation or direct angular velocities processed using Principal Component Analysis (PCA) [10] (see Fig. 6). This ability to effectively represent periodic motion data aligns with our objective to enhance the network's understanding of the robot's dynamics, also leading to improved odometry performance.

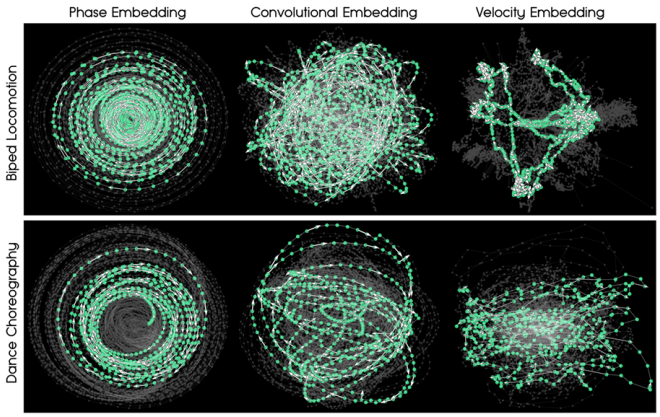


Fig. 6: 2D PCA embedding of the learned phase manifold compared to velocity and low-dimensional convolutional embeddings, highlighting the time series of a single motion clip of biped locomotion (top) and a dance choreography (bottom). (from [1])

V. Stationary Wavelet Transform

We chose to use the Wavelet Transform (WT) [11] because it provides both time and frequency information, unlike the FFT, which only operates in the frequency domain. Figure 7 illustrates the lack of temporal information when using the FFT.

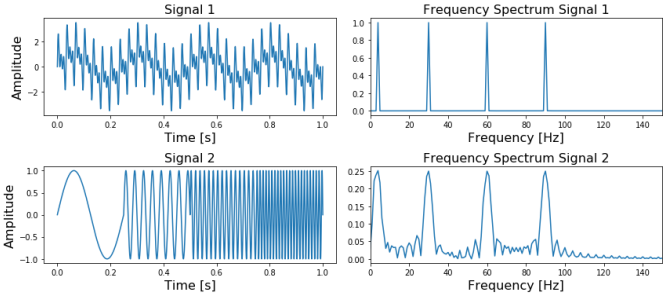


Fig. 7: Signals 1 (top) and 2 (bottom) contain four frequencies (4, 30, 60, 90 Hz). The four frequencies appear at all times for Signal 1 and sequentially for Signal 2. Even though the two signals are different, the frequency spectra on the right for both signals are similar.

The WT works by applying a sequence of low and high pass filters to the signal, where the filters use wavelets as their impulse response. Wavelets are localized oscillatory functions that help decompose the signal into various frequency components while retaining temporal information (see Fig. 8). This initial decomposition results in two types of coefficients:

- Approximation Coefficients (Low-Frequency Components): These are obtained by convolving the signal with a low pass filter.
- Detail Coefficients (High-Frequency Components): These are obtained by convolving the signal with a high pass filter.

The filtering process can be repeated at multiple levels. Each subsequent level captures progressively finer details.

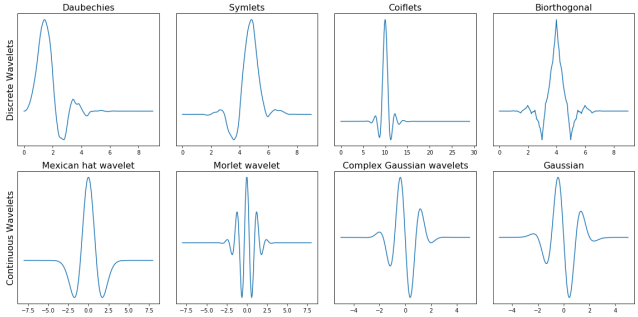


Fig. 8: Wavelet types

We use the Stationary Wavelet Transform (SWT) [2] because it maintains the same input and output shapes by avoiding the downsampling step present in the Discrete Wavelet Transform (DWT). We then use specific outputs from the SWT using the Daubechies 4 (db4) wavelet. Specifically:

- Level 3 coefficients for low-frequency information about the angular velocities (1 channel for each axis: x, y, z, totaling 3 channels).
- Level 1 coefficients for high-frequency information about the angular velocities (3 channels).
- Level 3 coefficients for low-frequency information about the linear acceleration (3 channels).

- Level 1 coefficients for high-frequency information about the linear acceleration (3 channels).

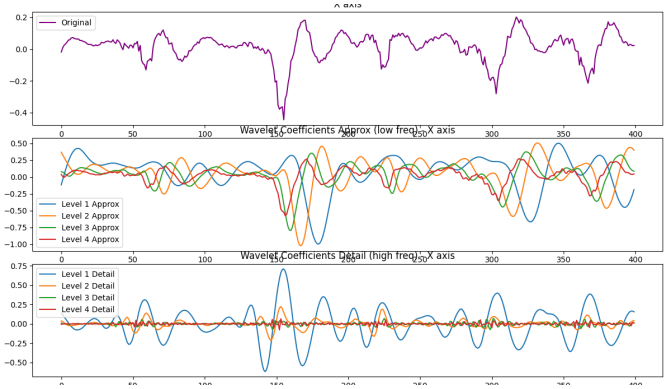


Fig. 9: Angular velocity of x axis (top), its SWT low frequency coefficients (middle) for multiple levels and its SWT high frequency coefficients (bottom) for multiple levels.

VI. Experiments

In this section, we evaluate the performance of our network for odometry using three key metrics introduced in [12]: Absolute Trajectory Error (ATE), Time-based Relative Trajectory Error (T_RTE), and Distance-based Relative Trajectory Error (D_RTE). These metrics are defined as:

- **Absolute Trajectory Error (ATE)**: The Absolute Trajectory Error (ATE) measures the difference between the estimated and ground truth trajectories by calculating the Root Mean Squared Error (RMSE) of their positions:

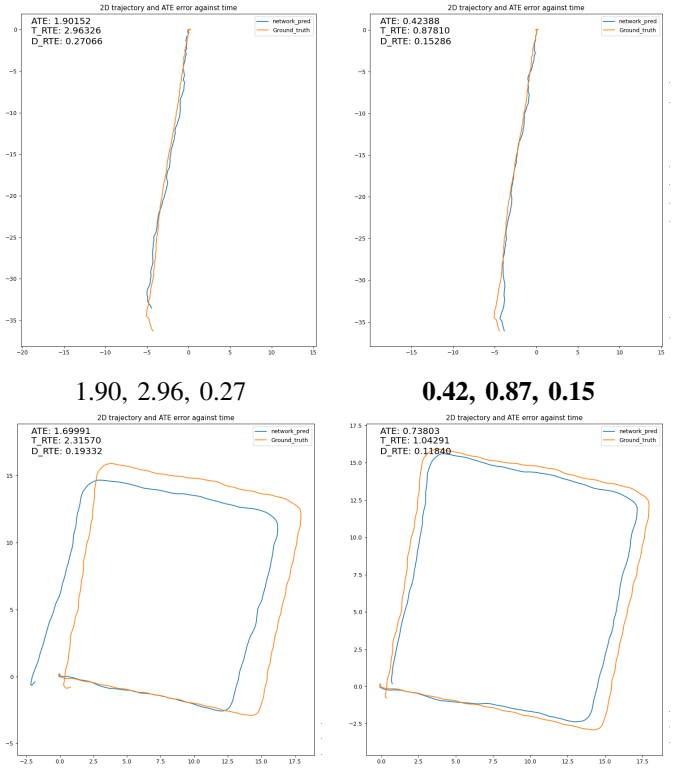
$$\text{ATE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{e}_i - \mathbf{g}_i\|^2} \tag{5}$$

  where $\mathbf{e}_i$ is the estimated position at time i, $\mathbf{g}_i$ is the ground truth position at time i, and n is the number of positions.
- **Time-based Relative Trajectory Error (T_RTE)**: The Time-based Relative Trajectory Error (T_RTE) evaluates the drift in the estimated trajectory over fixed time intervals. This metric computes the RMSE of the endpoint drifts from fixed windows of the trajectory, averaged over different window sizes. It measures how well the estimated trajectory aligns with the ground truth over various time segments, indicating the consistency of the odometry over time.
- **Distance-based Relative Trajectory Error (D_RTE)**: The Distance-based Relative Trajectory Error (D_RTE) evaluates the drift in the estimated trajectory over fixed distance intervals. Similar to T_RTE, it computes the RMSE of the endpoint drifts but does so over fixed distance segments, here of one meter. This metric assesses the accuracy of the trajectory estimation over spatial intervals, providing insight into how well the odometry maintains accuracy over varying distances.

The following results demonstrate that our network performs better and is able to predict 3D translations more accurately by introducing additional features and loss functions.

TABLE I: Comparison of performance metrics between RNIN and our network

Metric	RNIN	Our Network
Average ATE	3.0739	**2.3422**
Average T_RTE	3.7912	**2.8872**
Average D_RTE	0.2745	**0.1639**



1.90, 2.96, 0.27 **0.42, 0.87, 0.15**

1.69, 2.31, 0.19 **0.73, 1.04, 0.11**

Fig. 10: Results on unseen data from RNIN network (left) and ours (right). Numbers correspond respectively to ATE, T_RTE, D_RTE

VII. Motion Pattern Extraction

We extract features from a forward pass through the ResNet and use PCA [10] and t-SNE [13] to visualize these features in 2D. This helps us observe the inherent motion patterns of the agent (here a human), with similar motions clustering together as can be seen in Figure 12, indicating the network's understanding of underlying dynamics.

Using feature extraction from a neural network to learn about motion patterns can be particularly useful for robots with motions that are less understood, such as drones. For these robots, defining motion pattern thresholds is challenging due to their dynamic nature, whereas neural networks can infer these

patterns when properly trained. For example, **Neural-Fly** [7] leverages the observation that aerodynamics in different wind conditions share a common representation, with the wind-specific dynamics lying in a low-dimensional space. Applying this method to various robots can enhance their ability to understand and adapt to their motion patterns, improving navigation and interaction capabilities.

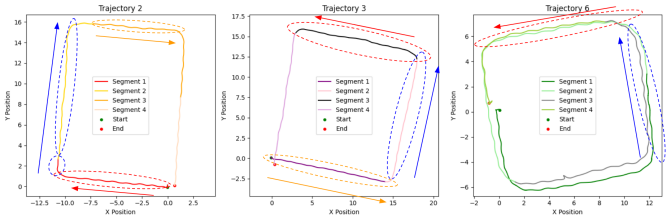


Fig. 11: 3 unseen trajectories split into 4 segments

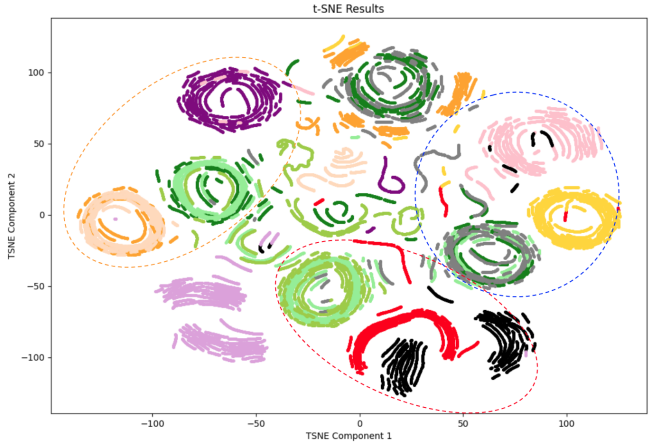


Fig. 12: t-SNE visualization of features from the 3 unseen trajectories in Fig. 11. Orange circled zone corresponds to right motion, blue zone corresponds to upward/straight motion, red zone corresponds to left motion.

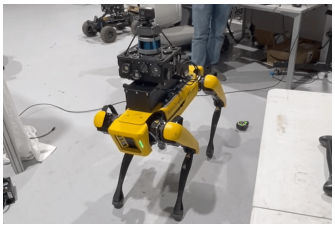
VIII. Future Work

In our future work, we aim to apply our approach to a wide variety of robots. We are currently working on this and have collected data from several robots in real life (see Fig. 13). We're also using NVIDIA's simulation and Reinforcement-Learning framework Isaac Lab [14] to collect data from robots, such as the humanoid robots H1 and G1, Anymal-C, Spot, and a quadcopter (see Fig. 14). Using simulation environments allows us to gather extensive data without the high costs associated with physical robots.

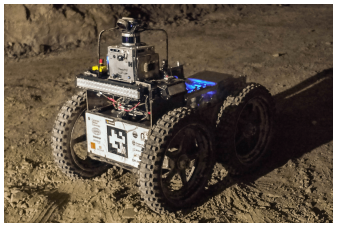
IX. Conclusion

In this project, we developed a learning-based approach for odometry that exclusively uses Inertial Measurement Unit (IMU) data. Our aim was not only to address the odometry task but also to enable the network to recognize and capture the intrinsic motion patterns of the agent. To accomplish this, we incorporated components such as Periodic AutoEncoders (PAE) and Stationary Wavelet Transforms (SWT) to extract significant features. Our results indicate that these enhancements significantly boost the accuracy and robustness of the odometry system and allow for the extraction of motion pattern information from the agent.

(a) UAV



(b) Spot



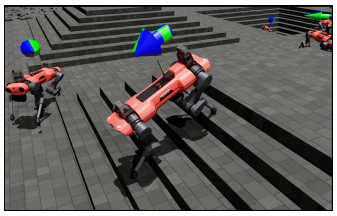
(c) UGV

(d) Off-road car

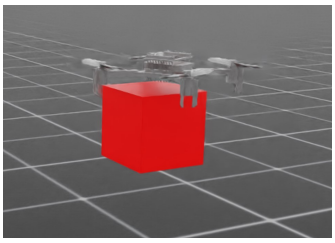
Fig. 13: Robots of the AirLab that we use data from.

(a) G1 humanoid

(b) Spot



(c) AnymalC



(d) Quadcopter

Fig. 14: Robots from Isaac Lab that we use data from.

References

[1] S. Starke, I. Mason, and T. Komura, "Deepphase: periodic autoencoders for learning motion phase manifolds," *ACM Trans. Graph.*, vol. 41, no. 4, jul 2022. [Online]. Available: https://doi.org/10.1145/3528223.3530178

[2] G. P. Nason and B. W. Silverman, *The Stationary Wavelet Transform and some Statistical Applications*. New York, NY: Springer New York, 1995, pp. 281–299. [Online]. Available: https://doi.org/10.1007/978-1-4612-2544-7$_1$7

[3] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1, 2004, pp. I–I.

[4] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 3565–3572.

[5] J. Zhang and S. Singh, "LOAM: lidar odometry and mapping in real-time," in *Robotics: Science and Systems X, University of California, Berkeley, USA, July 12-16, 2014*, D. Fox, L. E. Kavraki, and H. Kurniawati, Eds., 2014. [Online]. Available: http://www.roboticsproceedings.org/rss10/p07.html

[6] H. F. Grip, J. S. Lam, D. S. Bayard, D. T. Conway, G. Singh, R. Brockers, J. Delaune, L. H. Matthies, C. A. Malpica, T. Brown, A. Jain, A. M. S. Martin, and G. B. Merewether, "Flight control system for nasa's mars helicopter," *AIAA Scitech 2019 Forum*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:86511266

[7] M. O'Connell, G. Shi, X. Shi, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural-fly enables rapid learning for agile flight in strong winds," *Science Robotics*, vol. 7, no. 66, p. eabm6597, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6597

[8] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3D scene graph construction and optimization," 2022.

[9] D. Chen, N. Wang, R. Xu, W. Xie, H. Bao, and G. Zhang, "Rnin-vio: Robust neural inertial navigation aided visual-inertial odometry in challenging scenes," in *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2021, pp. 275–283.

[10] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901. [Online]. Available: https://doi.org/10.1080/14786440109462720

[11] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[12] S. Sun, D. Melamed, and K. Kitani, "Idol: Inertial deep orientation-estimation and localization," 2021.

[13] L. van der Maaten and G. E. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008. [Online]. Available: https://api.semanticscholar.org/CorpusID:5855042

[14] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, p. 3740–3747, Jun. 2023. [Online]. Available: http://dx.doi.org/10.1109/LRA.2023.3270034