Microscopic Marvels: Computer Vision for Miniature Robots

Aliya Tang and Elvina Wibisono, Computer Science
Brian Plancher, Computer Science

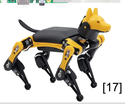
The Barnard Accessible and Accelerated Robotics Lab

The Big Picture

Our research optimizes visual motion estimation for tiny robots, focusing on the ***Petoi Bittle Robot Dog*** with ***Raspberry Pi 3 Model B*** and ***Raspberry Pi Camera V3***. Affordable hardware hosts efficient computer vision algorithms, enabling fully autonomous *tiny machine learning (tinyML)* [12] operations. Elevating accessible robots with cutting-edge vision opens new possibilities for diverse real-world tasks.

Key Features:

- **Comparative analysis of computational speeds** across embedded systems.
- **Evaluation of proposed algorithms** for efficient motion estimation.
- Envisioning a future of **highly capable and versatile tiny robots**.



[17]

Introduction

In robotics, locating and tracking tiny robotic dogs has gained immense significance for cost-effective solutions. Lightweight and affordable, these tiny robots are vital for on-ground disaster management, infrastructure monitoring, and emergency search and rescue [1], [5], [12]. *Raspberry Pi* and *Arduino* have emerged as the go-to embedded systems for such projects, striking a balance between affordability and computational power. Our research focuses on advancing computer vision capabilities in tiny robots, using the *Petoi Bittle Robot dog* as our testbed, powered by *Raspberry Pi 3 Model B* and *Raspberry Pi Camera V3*. Our objectives include ***implementing high-functioning computer vision algorithms in tiny robots, localization and mapping using a monocular camera, object detection and recognition, obstacle avoidance for enhanced mobility, and terrain mapping and classification***.

Embedded System Comparison:

Images	Embedded System	Price	Operating System	SoC	CPU Architecture	CPU Cores	CPU Clock Speed	GPU	RAM
	Raspberry Pi Zero 2 W	$15	Debian Linux	BCM2710A1	ARM Cortex-A53, (ARMv8, 64-bit)	4	1 GHz	VideoCore IV	512 MB
	Raspberry Pi 3 Model A+	$25	Debian Linux	BCM2837B0	ARM Cortex-A53, (ARMv8, 64-bit)	4	1.4 GHz	VideoCore IV	512 MB
	Raspberry Pi 3 Model B	$35	Debian Linux	BCM2837A0/B0	ARM Cortex-A53, (ARMv8, 64-bit)	4	1.2 GHz	VideoCore IV	1 GB
	NVDIA Jetson Nano	$149	Linux	NVIDIA Tegra X1	ARM Cortex-A57 MPCore Processor (ARMv8-A, 64-bit)	4	1.43 GHz	NVIDIA Maxwell	4 GB
	Arduino Nano ESP32	$13.70	N/A	u-blox® NORA-W106	ESP32-S3	2	16 MHz	N/A	512 kB
	Arduino Uno R4 Wifi	$27.50	N/A	Renesas RA4M1	Arm® Cortex®-M4	2	48 MHz	N/A	32 KB

Table 1. Comparisons of Computational Power in Different Embedded Systems

We compared various embedded systems' computational capabilities and **selected Raspberry Pi 3 Model B**. We focus on integrating the complex VSLAM algorithm into our tiny computer. By successfully implementing VSLAM, we **enhance the robot's capabilities, bringing cost-effective, high-performance computer vision to various applications**.

Background

Fully autonomous applications require the robot to precisely measure its location in the listed situations. Two categories of visual motion estimation methods were explored: feature-based methods and direct methods. Feature-based algorithms extract features from images and matching them in successive frames using feature descriptors, stabilizing camera motion, and aligning the structures through reprojection error minimization [5]. Many **feature detectors are optimized for speed rather than precision; thus, a drift in motion can lead to estimation errors**. On the other hand**, direct methods rely on the intensity values in the image** to estimate the structure and motion of the tiny robot [5]. The photometric error intensity of direct methods is much more powerful than the reprojection error of feature-based methods. The medium lies in between, which is the main focus of this research: **optimizing advanced computer vision algorithms to fit low computational power computers for tiny robot machine learning.** Both visual motion estimation methods minimize different errors; therefore, combining the two methods achieves precise, and low-cost results.

Some of the visual motion estimation algorithms include optical flow, VSLAM, and Visual Odometry (VO). Each algorithm differs from the others; for example, VSLAM encompasses localization and mapping, while VO only estimates the camera's motion from images [3], [14], [7]. However, both use sensors such as cameras or inertial measurement units. After a thorough background literature review, **VSLAM algorithms were more suitable** for our implementations of visual motion estimation methods because they utilize both localization and mapping.

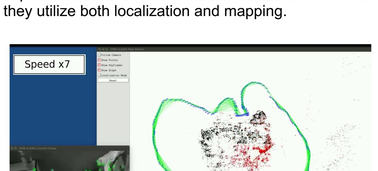


Fig. 4. ORB SLAM Localization and Tracking Map [14]

Fig. 5. TDK 6-Axis IMU

Methods

Our research focuses on visual motion estimation algorithms for tiny robots using Raspberry Pi hardware. We began by individually *annotating papers to identify challenges* in computer vision algorithms and *key components of visual motion estimation*. **Setting up the hardware, including Raspberry Pi 3 Model A+ and B with Pi cameras, was our initial step**.

Paper Name	Year	Direct	Featured	Monocular	Open Source Code	# of Dependencies
LIFT-SLAM: A deep learning feature based monocular visual SLAM method [10]	2021		X	X		
CubeSLAM: Monocular 3D Object SLAM [16]	2019		X	X	X	3
DynaSLAM: Tracking, Mapping and Inpainting in Dynamic Scenes [2]	2018		X	X	X	9
ORB-SLAM: a Versatile and Accurate Monocular SLAM System [14]	2015		X	X	X	5
PL-SLAM: Real Time Monocular Visual SLAM with Points and Lines [1]	2017		X	X	X	8
FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks [8]	2016	X		X	X	8
Parallel Tracking and Mapping for Small AR Workspaces [9]	2007		X	X	X	4
OpenVSLAM: A Versatile Visual SLAM Framework [15]	2023		X	X		
Learning by Inertia: Self-Supervised Monocular Visual Odometry for Road Vehicles [6]	2019	X		X		
SVO: Fast Semi-Direct Monocular Visual Odometry [5]	2017	X	X	X	X	5

Table 2: Example of Filtering Process

Algorithm Testing:

- Utilizing *OpenCV*, *Edge Impulse*, and *TensorFlow*, we tested simple **object detection** algorithms on Raspberry Pi 3 Model A+.
- From our literature review (Table 2), **we selected specific algorithms for implementation**, exploring their efficiency on low computational power computers.

Algorithm Implementation:

1. **Visual Simultaneous Localization and Mapping (VSLAM):** Attempted to implement VSLAM algorithms, optimizing the code for energy efficiency on Raspberry Pi 3 Model A+ and B.
2. **Semi Direct Visual Odometry (SVO):** We also explored implementing SVO, striving for code optimization to ensure efficient performance on the chosen Pi models.

Deployment Results & Discussion

VSLAM Implementation Challenges:

- Raspberry Pi 3 Model A+ and B **lacked computational power** to download dependencies and run VSLAM code.
- Explored specific VSLAM algorithms, including **ORB-SLAM** [14] and **OpenVSLAM** [15], but faced **hardware issues and interdependent code complexities**.

Adapted Solution:

- ***MicroROS Integration*:** Installed microROS, a modified ROS variant for resource-constrained microcontrollers .

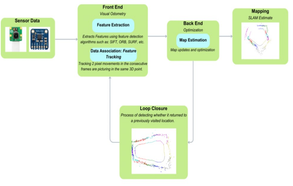


Fig. 1. Traditional vSLAM algorithm

Semi Direct Visual Odometry (SVO) is a promising VO algorithm that **combines direct and feature-based methods**. This fusion allows for precise identification of salient features in diverse textured environments, ensuring more accurate results. In *low-textured* settings, ***direct methods*** excel, while ***feature-based methods*** perform better in *high-textured environments* [5]. Compared to the traditional vSLAM algorithm (Fig.1.), SVO does not perform *loop closure or bundle adjustment*, which makes the algorithm to be well suited on systems with low computational power

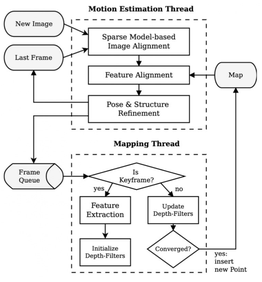


Fig. 2. SVO Algorithm Architecture [5]

SVO Implementation and Hardware Optimizations:

- **Implemented SVO without ROS**, modifying deprecated packages using CMake for Raspberry Pi compatibility.
- **Manually adapted and integrated sparse model image alignment** files from SVO code to suit Raspberry Pi 3 Model B.
- **Raspberry Pi 3 Model B chosen over Model A+ for improved performance** (Table 1).

Conclusions & Next Steps

Implementing computer vision algorithms on small computers for tiny robots is feasible, but it comes with **software and hardware challenges**. Time constraints and code modifications for Raspberry Pi limited our progress. Despite the Raspberry Pi's low computational power, we aim to enable inexpensive tiny robots for emergency situations, offer students access to affordable computer vision strategies, and aid researchers in optimizing advanced algorithms for minimal sensor setups. We are currently implementing a modified version of Semi Direct Visual Odometry (SVO) on Raspberry Pi.

By pushing the boundaries of what is possible with affordable hardware, we envision a future where tiny robots **play an integral role in various communities**:

- **Engineers** can harness inexpensive tiny robots for emergency response, efficiently mitigating critical situations.
- **Students** gain access to affordable computer vision strategies, fostering learning and innovation in robotics.
- **Researchers** can further optimize advanced computer vision algorithms for low-cost tiny robots equipped with minimal sensors.

Next Steps:

1. **Develop Our Own Modified SVO**: Optimize the SVO code for Raspberry Pi, enhancing computational efficiency.
2. **Performance Comparison**: Compare advanced algorithms on Raspberry Pi and Jetson Nano, using graphs and FPS measurements.
3. **Hardware Identification**: Select suitable hardware for further exploration of complex CV algorithms.
4. **Integration of Additional Methods:** Incorporate findings from research papers into the modified SVO code.
5. **Extensive Reading**: Identify key areas for further development in tiny robot computer vision.
6. **Expand Algorithm Implementation**: Implement Object Detection, Object Avoidance, Terrain Mapping, and more after successful SVO implementation.
7. **Engaging Demos:** Create compelling demos showcasing the potential of our modified computer vision algorithms.

References

1. A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu and F. Moreno-Noguer, "PL-SLAM: Real-time monocular visual SLAM with points and lines," 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017, pp. 4503-4508, doi: 10.1109/ICRA.2017.7989522.
2. B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "DynaSLAM: Tracking, Mapping, and Inpainting in Dynamic Scenes," in IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 4076-4083, Oct. 2018, doi: 10.1109/LRA.2018.2860039.
3. Bai, Y., Zhang, B., Xu, N., Zhou, J., Shi, J., & Diao, Z. (2023). Vision-based navigation and guidance for agricultural autonomous vehicles and robots: A review. Computers and Electronics in Agriculture, 205, 107584.
4. B. Duisterhof, S. Krishnan, C. Banbury, E. L. Colombini, H. M. S. Bruno, and S. M. Neuman, "LIFT-SLAM: A deep-learning feature-based monocular visual SLAM method," Neurocomputing, vol. 455, pp. 97-110, Sep. 2021, doi: 10.1016/j.neucom.2021.05.027.
5. C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 2014, pp. 15-22, doi: 10.1109/ICRA.2014.6906584.
6. C. Wang, Y. Yuan, and Q. Wang, "Learning by Inertia: Self-supervised Monocular Visual Odometry for Road Vehicles," ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 2019, pp. 2252-2256, doi: 10.1109/ICASSP.2019.8683446.
7. D. Scaramuzza and F. Fraundorfer, "Visual Odometry [Tutorial]," in IEEE Robotics & Automation Magazine, vol. 18, no. 4, pp. 80-92, Dec. 2011, doi: 10.1109/MRA.2011.943233.
8. E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 1647-1655, doi: 10.1109/CVPR.2017.179.
9. G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 2007, pp. 225-234, doi: 10.1109/ISMAR.2007.4538852.
10. H. M. S. Bruno and E. L. Colombini, "LIFT-SLAM: A deep-learning feature-based monocular visual SLAM method," Neurocomputing, vol. 455, pp. 97-110, Sep. 2021, doi: 10.1016/j.neucom.2021.05.027.
11. J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars (Eds.), Computer Vision – ECCV 2014, ECCV 2014, vol. 8690 of Lecture Notes in Computer Science, Springer, Cham, 2014, pp. 834-849, doi: 10.1007/978-3-319-10605-2_54.
12. J. Jabbour, M. Mazumder, B. Plancher, V. J. Reddi, S. Krishnan, C. Banbury, S. Prakash, S. M. Neuman, and B. Duisterhof, "Tiny Robot Learning: Challenges and Directions for Machine Learning in Resource-Constrained Robots," in 2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS), Incheon, Korea, Republic of, 2022, pp. 296–299.
13. R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in 2011 International Conference on Computer Vision, Barcelona, Spain, 2011, pp. 2320-2327, doi: 10.1109/ICCV.2011.6126513.
14. R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," in IEEE Transactions on Robotics, vol. 31, no. 5, pp. 1147-1163, Oct. 2015, doi: 10.1109/TRO.2015.2463671.
15. S. Sumikura, M. Shibuya, and K. Sakurada, "OpenVSLAM: A Versatile Visual SLAM Framework," in Proceedings of the 27th ACM International Conference on Multimedia (MM '19), Nice, France, 2019, pp. 2292-2295, doi: 10.1145/3343031.3350539.
16. S. Yang and S. Scherer, "CubeSLAM: Monocular 3-D Object SLAM," in IEEE Transactions on Robotics, vol. 35, no. 4, pp. 925-938, Aug. 2019, doi: 10.1109/TRO.2019.2909168.
17. Images retrieved from official Raspberry Pi and Bittle websites

Acknowledgements

We would like to thank the Arthur Vining Foundation and Barnard College for their funding and support. We would like to thank Professor Brian Plancher for being a great mentor. We would like to thank the Science Research Institute for the opportunity to showcase our work.