ARMS: Automated Rover for Mobile Support

Omri Steinberg-Tatman
EEC 174BY
College of Engineering
Davis, CA
ost@ucdavis.edu

Jackson Vaughn
EEC 174BY
College of Engineering
Davis, CA
jvaughn@ucdavis.edu

Brooke O'Flaherty
EEC 174BY
College of Engineering
Davis, CA
kelpiekrisps@gmail.com

Luke Jones
EEC 174BY
College of Engineering
Davis, CA
lpjones@ucdavis.edu

***Abstract*—This paper introduces the Automated Rover for Mobile Support (ARMS), an innovative solution designed to alleviate the physical burden of carrying heavy backpacks for students. ARMS is capable of autonomously following its owner while carrying essential items, utilizing a modified Traxxas RC car equipped with an NVIDIA Jetson TX2 module and an Intel RealSense camera. Leveraging NVIDIA's trt_pose model for real-time pose estimation, the rover interprets specific user poses for control, enabling hands-free interaction that enhances the rover's convenience and usability. The implementation includes a detailed analysis of the hardware setup, pose detection accuracy, and user-following functionality. Our system achieves an operational frame rate close to real-time, ensuring responsive and reliable rover behavior.[1].**

Additionally, through rigorous real-world testing, ARMS demonstrated the capability to navigate diverse environments, respond to dynamic user commands, and safely transport items with minimal user input. The paper also discusses potential future enhancements—including variable following distance and multi-object tracking and segmentation (MOTS)—to expand the rover's applicability and performance further. The success of ARMS highlights the potential of autonomous robotic solutions in supporting physical daily activities, particularly in educational settings, by reducing physical strain and improving the quality of life for students.

***Index Terms*—machine learning, pose detection, autonomous driving**

I. Introduction

In the United States, over 79 million students rely on backpacks to transport their daily essentials. The necessity for carrying many items—particularly heavier electronics—for daily educational needs has become a common practice. However, this trend is taking a toll on students' physical health; while experts recommend that a backpack not exceed 15% of the wearer's total body weight, nearly 55% of students are burdened with bags surpassing their recommended carrying capacity. This over-burdening led to approximately 23,000 backpack-related injuries in 2007 alone. [1]

Unfortunately for their backs, it is not possible for students to simply carry fewer items. Students need some helping hands, or, more aptly, some helping arms. ARMS—or Autonomous Rover for Mobile Support—is our solution to the problem weighing on everyone's shoulders: backpacks. By repurposing a small Traxxas RC car and equipping it with an NVIDIA Jetson TX2 module and an Intel RealSense camera, we've created a rover that autonomously follows its user from one class to another, shouldering the load of their most essential items. Additionally, ARMS allows hands-free interaction for users through specific poses, thanks to NVIDIA's trt_pose model for real-time pose estimation, offering a convenient and efficient way to manage the rover's operations during daily activities.

[1]Our complete GitHub repository can be found at https://github.com/Just-Jacksone/ARMS

II. Rover Hardware

The rover itself is vital to the functionality of ARMS and each of its components plays an important role. This section will serve as an outline of the hardware implementation of the rover and the steps required for setting it up.

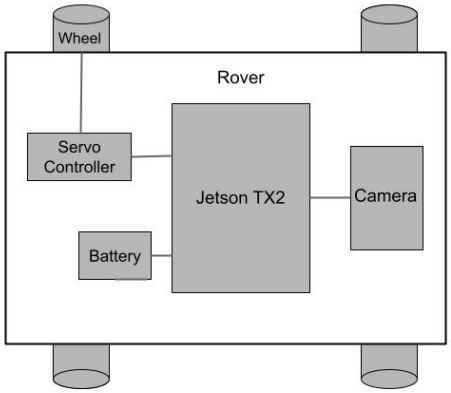

Fig. 1. Layout of the Rover

The diagram pictured in Figure 1 abstracts how ARMS's components work together. The NVIDIA Jetson TX2 module is the central processor, handling the computations, pose estimation, and communication between components; the Intel RealSense camera acts as the rover's vision, capturing video of the user and using its LiDAR sensor to measure the distance for automatic, emergency braking; finally, the servo controller enables the Jetson TX2 to steer the rover by controlling the wheels and the speed through the electronic stability control (ESC).

III. Pose Detection

A. Pose Design

Users command the rover through specific poses. To initiate movement, users get into the T-pose—an upright stance with arms extended straight horizontally. To recognize these poses, we employed NVIDIA's trt_pose, a tool designed for real-time pose estimation on the Jetson board. trt_pose comes with models pre-trained for human pose estimation, which we used directly; it interprets poses using the Microsoft Common Objects in Context (MS COCO) format, where each body joint is assigned a unique numerical identifier—such as assigning the number 5 to the right elbow. This format facilitates data extraction for each body joint necessary for pose detection.

trt_pose analyzes individual frames captured by the camera. Each frame is processed to extract the pose information, specifically focusing on the raw coordinates of key body joints. We stored these coordinates in a 'Body' class, focusing on the elbows, wrists, shoulders, and chest positions essential for our arm-dependent poses. We determined the user's pose by comparing the relative joint coordinates of these key points. As they were not used in our key poses, there was no need to track leg positions.

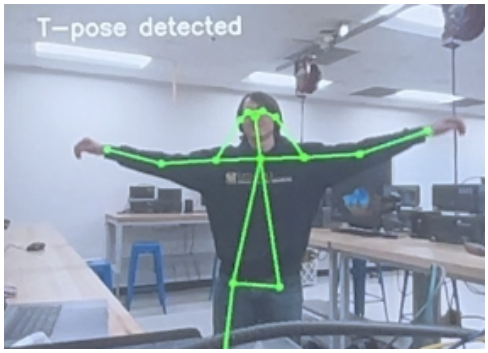

Fig. 2. T-Pose detected using trt_pose

B. Pose Implementation

The rover recognizes a T-pose when the user's wrists, shoulders, and elbows align horizontally, with the joints' y-levels falling within a specific range of each other. We adjusted this range to suit our team's needs, but individual calibration might be needed for different users. Similarly, an S-pose is detected when the elbows and chest are on the same horizontal plane, with one wrist positioned higher and the other lower than this plane. This setup allows for a variety of poses to be used for control. During testing, we also implemented 'W' and 'M' poses for adjusting the rover's speed, where users position both their wrists either above or below the chest level for 'W' and 'M' poses respectively.

Each pose triggers a specific command, with the T-pose serving as a start and S-pose as a stop. The T-pose activates the program and signals the rover to start tracking the user. The S-pose pauses the program, prompting it to reset to an idle state and search for a T-pose to begin tracking again. During our trials, we found that the additional poses could be used to modify the rover's speed or following distance. These pose-based instructions offer users intuitive, direct control over the rover's movements and operational states.

IV. User Following

A key functionality of ARMS is its ability to follow users autonomously, achieved through processing data from the trt_pose framework. Unlike typical applications of trt_pose for pose estimation, our utilization locates the user's chest node, using it as a dynamic reference point to represent the user's position relative to the rover.

The system retrieves the chest node's coordinates in each operational cycle to compare against the camera's central axis. The discrepancy between these two points informs the adjustment of the rover's steering, ensuring the user remains centered in the camera's field of view. For example, if the user is detected to the far left of the center coordinate, the rover's wheels will turn entirely left; similarly, a central alignment results in straight wheel orientation when the user is centered with the camera. This reference-based steering method guarantees accurate direction adjustments for the rover to follow the user. The conversion factor we found to work best can be seen below:

$$x = (0.28125 \times x) + 90 \tag{1}$$

The depth information provided by the chest node—obtained through the camera's depth-sensing capabilities—plays a crucial role in maintaining an optimal following distance. By continuously monitoring the distance between the rover and the user, the rover advances towards the user as long as they remain beyond a predetermined, safe threshold distance. Conversely, should the user approach too closely or exit the camera's view—indicating they are too close or untrackable—the rover halts to avoid potential collisions. This safety mechanism ensures that the rover maintains a safe distance from the user, stopping movement when the user's position is uncertain or too near, thereby minimizing the risk of injury.

V. Pipeline Design

The full pipeline of the rover is pictured in Figure 3.

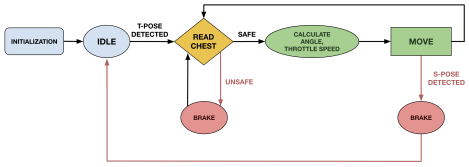

Fig. 3. Automated Pipeline

The pipeline begins at the 'INITIALIZATION' stage by turning on the rover and executing an initialization sequence to begin communication between the Jetson TX2 module and its externally connected components; specifically, this sequence begins communication between the module and the servos and motor of the rover, and RealView camera. This sequence also imports all relevant packages and other startup items. Once initialized, the rover enters an 'IDLE' state where it scans for an on-screen T-pose. While no T-pose is detected, the rover remains stationary. Once a T-pose is detected, the rover enters its main logic loop of reading the coordinate and depth value of the user's chest node—coordinates determined by trt_pose and depth by the LiDAR sensor of the RealView camera.

Then, as pictured in the green portion of the pipeline, the rover calculates the angle and throttle to move toward the user if safe and finally moves towards the user. The rover only exits this logic loop upon detecting an S-pose, wherein it brakes and speed is set to 0 to stop motion and returns to the 'IDLE' state; however, if—after reading the depth of the chest node—the user is found to be closer than the safe-distance threshold, or no user is found on the screen the rover will break and stop movement until the user has moved the threshold of distance away or the user has come back into the camera view. These automatic braking features prevent jeopardizing user safety.

Once the rover's algorithm determines the user is 'UNSAFE' to follow, it continuously reads the chest position until the user is found in a safe position, resuming the main loop's operation. To calculate the steering angle, the algorithm compares the horizontal position of the chest against the center point of the camera; it then finds the turn angle—between a full left, straight forward, and a full right—proportionate to the distance. Moreover, if a user is fully to the camera's left, the wheels will turn to make a maximum left; if the user is in the center, the wheels will adjust to drive the rover straight; and if the user is fully to the camera's right, the wheels will turn to make a maximum right.

VI. Benchmarks

Our system must operate with minimal lag to minimize the delay between the user's movements and the rover's response. The goal is for the rover to process and react to inputs at a rate that approaches real-time, ideally at around 24 frames per second (FPS). Although our rover operates slightly below this target—achieving around 20 FPS—this performance is close to our ideal rate. At this speed, the rover responds to user poses and movements quickly enough to maintain a smooth and responsive operation.

Examining trt_pose's performance is crucial for understanding the rationale behind its selection. While NVIDIA provides limited specific performance data for trt_pose, they offer throughput FPS for various models, which is informative for our purposes. Notably, they provide the performance of trt_pose on a model like resnet18, which is relevant to our hardware setup on the NVIDIA Jetson TX2, indicating an expected throughput of approximately 22 FPS. This figure aligns well with our operational goals, supporting the decision to consider trt_pose as a viable option for real-time pose estimation. While not identical, the comparison to the Jetson Nano suggests that the Jetson TX2 can achieve similar—if not slightly better—performance, reinforcing the feasibility of achieving near-real-time responsiveness in our project.

Model	Jetson Nano	Jetson Xavier
resnet18_baseline_att_224x224_A	22	251
densenet121_baseline_att_256x256_B	12	101

TABLE I
TRT_POSE FPS PERFORMANCE

VII. Real-World Testing

To validate the performance of our rover and ensure it meets design specifications, thorough testing in dynamic, real-world conditions was essential. Although the initial development and debugging were conducted while the rover was stationary, evaluating its functionality in a live environment is crucial for confirming its practical effectiveness. Our testing focused on the rover's ability to:

1) Follow the user at a reasonable pace.
2) Navigate around obstacles while maintaining pursuit of the user.
3) Execute 90-degree turns to keep up with the user.
4) Halt to avoid collisions with the user and others.

For our testing we utilized the hallways of Kemper Hall at the University of California, Davis. This location was selected for its convenience and because it represents an academic environment where ARMS is intended to be used. With its extended, winding corridors, the Kemper Hall lobby provided an ideal testbed for assessing the rover's agility and follower capabilities in tight spaces.

Although our rover achieved its primary goals, we recognized several opportunities for enhancement. A notable challenge emerged when the rover operated in environments with multiple individuals, negatively affecting its performance in public spaces. Furthermore, the rover tended to excessive micro-adjustments, likely attributed to camera instability and the lack of direct communication between steering and throttle mechanisms. This behavior may also stem from the natural movement patterns of users, suggesting that extended tuning could mitigate these issues. While the rover could execute tight turns, there was a discernible error margin, necessitating greater attention from the user. Although adjustments could be made to slow down turns, achieving flawless execution remained elusive. Despite these shortcomings, the rover successfully fulfilled its essential functions. We will delve into these challenges and potential solutions in the forthcoming section on future enhancements.

VIII. Future Improvements

Although we have put a full quarter of effort from our whole team into the rover, there is still significantly more we want to continue building.

A. Variable Following Distance

Throughout this report, we have highlighted our rudimentary approach to establishing a following distance through fixed measurement of the chest node's depth for ARMS. To enhance the functionality and adaptability of our rover, we propose introducing a dynamic following distance mechanism. This innovation would utilize the tracking setup at the beginning of the program—triggered by a T-pose—to set the user's baseline following distance. Thereafter, users could adjust this distance as desired through specific gestures: a W-pose to increase the distance, making the rover follow further behind, and an M-pose to decrease it, allowing the rover to close in. This adjustment capability necessitates the rover to operate at a range of speeds rather than a fixed set, ensuring it accelerates to maintain or decelerates to preserve the desired following distance.

Implementing such a feature involves integrating a PID (Proportional, Integral, Derivative) control system. Although we explored this early in the project, we deemed it beyond our current scope due to the need for extensive calibration to prevent feedback loops that could negatively affect the rover's steering and speed control. Additionally, adjustments for the camera's positioning and stabilization may be required to accommodate the user's height and ensure a consistent viewing angle without interference.

Incorporating a variable following distance would significantly enhance the rover's usability in various scenarios. For instance, increasing the distance is particularly beneficial at faster speeds or when the rover carries heavy loads, providing an additional safety margin for sudden stops. Conversely, a reduced following distance is advantageous in confined spaces or crowded environments, ensuring the user remains prominently in the camera's view, facilitating better navigation and tracking.

B. Multi-Object Tracking & Segmentation (MOTS)

Integrating Multi-Object Tracking and Segmentation (MOTS) would exponentially increase the rover's capabilities in crowded public spaces. Our current system is limited to recognizing a single figure, which poses challenges in environments with multiple people, reflections, or similar distractions. We envisioned establishing user recognition through the T-pose initialization, with subsequent tracking managed by advanced algorithms capable of distinguishing the designated user against other figures under various conditions. While we considered leveraging trt_pose's tracking capabilities, time constraints prevented us from fully implementing this feature. An alternative approach could involve combining trt_pose with models like YOLO for enhanced tracking precision, focusing on specific trt_pose outputs to accurately identify and follow the user.

The potential applications for a rover equipped with MOTS and dynamic following distance are vast and diverse. The implications are far-reaching, from transporting heavy school backpacks or luggage through urban settings, to facilitating the movement of oversized items in offices or factories. This technology could revolutionize how we move items in crowded spaces, navigate public transport, and even assist in personal shopping experiences by carrying purchases. Ensuring the optimal-height positioning of the camera and its secure attachment to the rover will maximize these advancements.

Our rover project broadens its practical applications by addressing these enhancements and sets the stage for future personal and commercial mobility solutions innovations.

IX. Conclusion and Discussion

A. Conclusion

Our project, the Automated Rover for Mobile Support—or 'ARMS'—demonstrates the successful integration of robotics and machine learning to alleviate the physical strain of carrying heavy loads, such as student backpacks. By repurposing a Traxxas RC car with advanced components like the NVIDIA Jetson TX2 module and an Intel RealSense camera, we created a rover capable of autonomously following its user. Leveraging NVIDIA's trt_pose for real-time pose estimation enabled intuitive, hands-free control over the rover, enhancing its usability and convenience. Our system achieved near-real-time performance, operating at an average of 20 frames per second, which, while slightly below our target, proved sufficient for a responsive and reliable user following. Throughout our real-world testing, ARMS consistently demonstrated the ability to navigate varied environments, respond to dynamic user commands, and safely transport items, showcasing its practical application in educational settings. The project's success opens avenues for future advancements in autonomous support systems, potentially transforming how students and others manage physical burdens in daily life.

B. Discussion

The journey of developing ARMS highlighted several key learning points and areas for further improvement. For starters, the project underscored the importance of real-world testing in robotics, revealing the challenges of environmental navigation and dynamic interaction not apparent in ideal, simulation-based testing. Next, the practical application of machine learning through NVIDIA's trt_pose model for pose detection illustrated the balance between computational efficiency and real-time responsiveness necessary for autonomous systems. Looking forward, we identified potential enhancements that could significantly extend ARMS's functionality and applicability. A variable following distance would allow for more adaptive user following, especially in crowded or confined spaces. Moreover, integrating Multi-Object Tracking and Segmentation (MOTS) would enable ARMS to navigate environments with multiple people more effectively, reducing the likelihood of tracking errors and improving overall system robustness. Despite the project's successes, we encountered limitations, such as the fixed following distance and the current system's reliance

on specific user poses for control. While functional, these aspects could benefit from further refinement to enhance user experience and system adaptability. In conclusion, the development of ARMS has been a comprehensive learning experience, combining elements of hardware engineering, software development, and practical implementation. The project achieved its primary goal of reducing the physical burden on students and provided valuable insights into the complexities of autonomous system design. As we continue to build on this foundation, we look forward to exploring new technologies and methodologies to bring our vision of supportive robotics closer to reality.

APPENDIX

This project was a collaborative effort by Jackson Vaughn, Luke Jones, Brooke O'Flaherty, and Omri Steinberg-Tatman. All programming, design, testing, and debugging activities were conducted jointly in the lab. Despite this collaborative approach, specific tasks were allocated among team members. Brooke O'Flaherty and Jackson Vaughn led the initial hardware setup, preparing the car for programming. Omri Steinberg-Tatman and Luke Jones undertook the preliminary setup of trt_pose on a non-Jetson computer, providing valuable insights into its application. Apart from these distinct tasks, the project was a unified group endeavor.

ACKNOWLEDGMENT

We extend our gratitude to Professor Chuah for her invaluable guidance throughout this project. We also wish to thank Kartik Patwari for his technical support and the ECE Department of The University of California, Davis, for providing the necessary funding. Their collective support was instrumental in the realization of ARMS.

REFERENCES

[1] Hospital, Huntsville. Back-to-School Burden: 79 Million Students in U.S. Carry Backpacks. Advance Local, 9 Aug. 2011, www.al.com/living/2011/08/back-to-school_burden_79_milli.html.

[2] Nvidia-Ai-Iot. "Nvidia-Ai-IOT/TRT_POSE: Real-Time Pose Estimation Accelerated with Nvidia TENSORRT." GitHub, github.com/NVIDIA-AI-IOT/trt_pose. Accessed 9 Mar. 2024.

[3] "Jetsonhacks - Overview." GitHub, github.com/jetsonhacks. Accessed 9 Mar. 2024.

[4] "Real-Time Human Pose Estimation." NVIDIA Developer, developer.nvidia.com/embedded/community/jetson-projects/nv_trt_pose. Accessed 9 Mar. 2024.