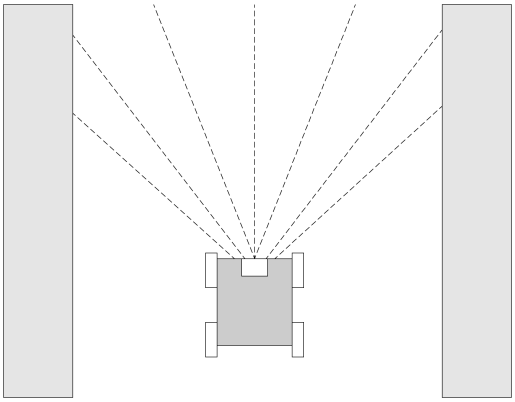
Team 1 Technical Report

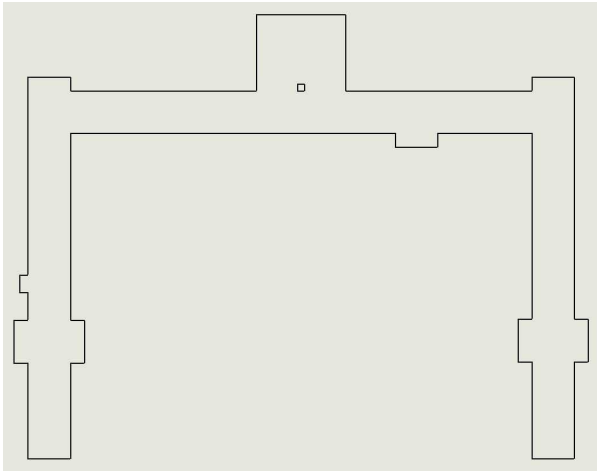
Davis Landry, * Earl Potters, Zak Laouar, † Jamison McGinley, ‡ Jason Evarts, § Tim Euken,¶
University of Colorado - Boulder

I. Introduction

The objective of this project is to design and create a $\frac{1}{10}$ scale autonomous robotic vehicle with the ability to navigate a course at speed and attempt challenges. The robotic vehicle navigated the known course in 4 separate heats, two clockwise, two counter-clockwise, for competitive time.



(a) Basic Vehicle Configuration *Proposed mounting and view angles of the Intel RealSense D435i*



(b) Sparse Course Map *Proposed map of the challenge course*

Figure 1: Illustrations of the setup and proposed course

Fig. 1b shows the course that was navigated for this project. With prior knowledge of the general layout of the course, tuning key parameters in the navigation control was both achievable and necessary. The challenges range from A-type, being the most difficult, to C-type which are the easiest. As a team, the challenges we chose to attempt were two B-type: stopping at a stop-sign and creating a sparse map while racing. To achieve these aims, the vehicle could be outfittrd with an Odroid XU-4 for all computations and sensors in the form of a D435i Intel RealSense camera for RGB-D data and a Phidget Spatial 3/3/3 Basic 1042 Inertial Measurement Unit. The vehicle only uses the chassis, battery, compute, and sensing equipment, wires, and other connecting equipment provided by the class.

II. Methodology

To solve the proposed challenge, a number of sub problems were solved including hardware, sensor input, and software based logic. The combination of these subsystems was integrated into the final autonomous system.

*106022964
†106319283
‡105207291
§105442284
¶109074757

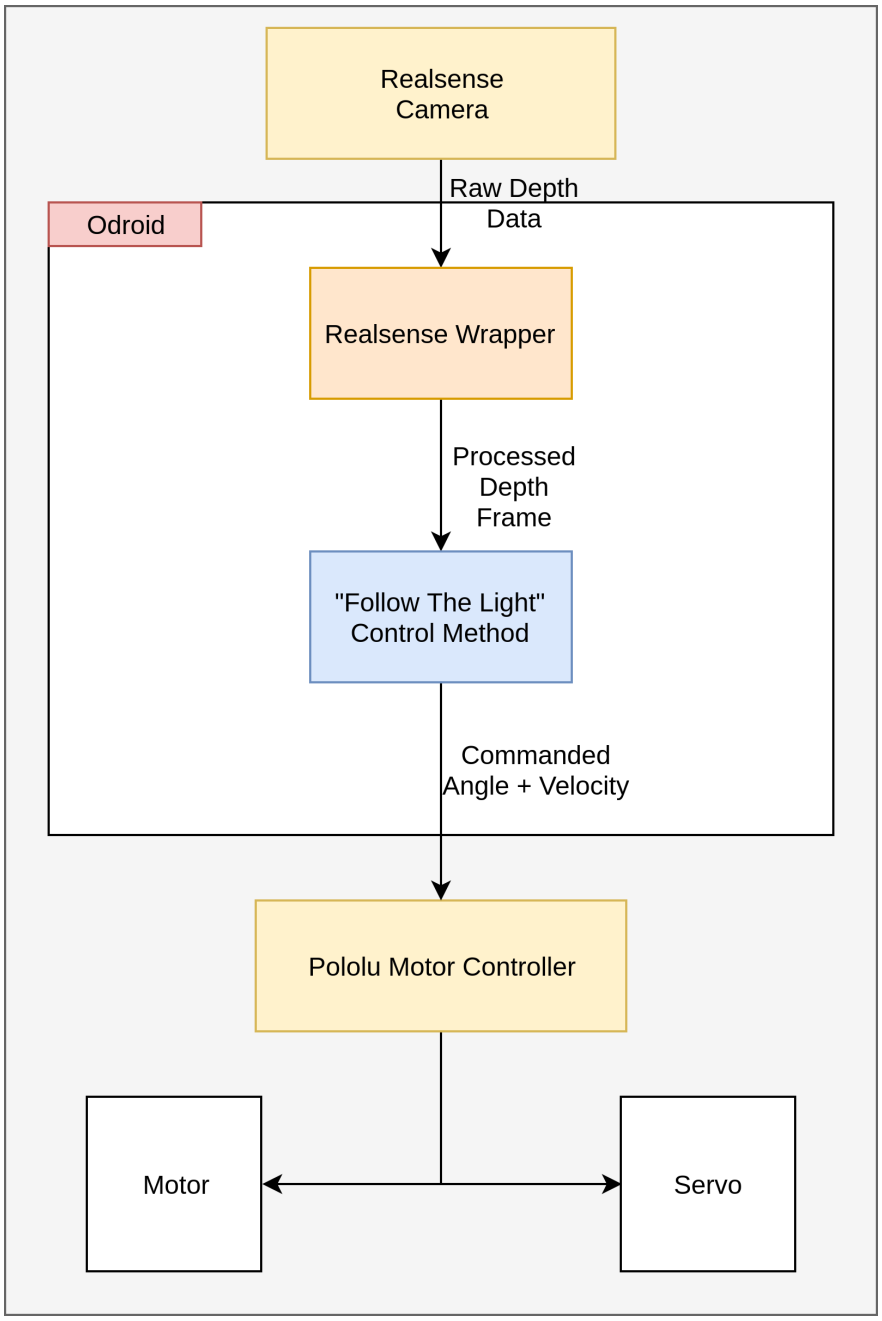


Figure 2: Functional Block Diagram of System

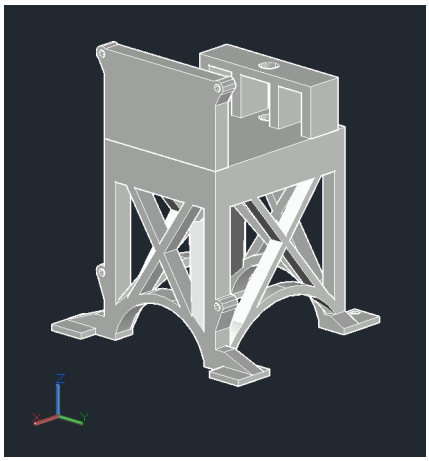
A. Hardware

The hardware solution was largely a problem of how to mount all of the components required for the robot to function within the $\frac{1}{10}$ AMP MT 2WD Monster Truck RTR chassis provided. The components that needed to be mounted include: ODROID XU-4, Intel RealSense D435i, pololu Maestro Servo controller, voltage regulator, Electronic Speed Controller (ESC), and batteries for operation of both the ODROID and the drivetrain.

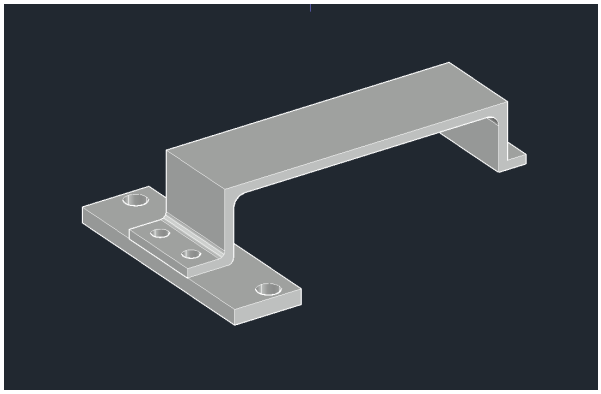
The first discussions we had were about the mounting of the Intel RealSense camera because poor placement of the camera would cause lots of problems based on improper localization. We wanted the camera to be close to the front of the car to maximize our view of the track ahead without the obstruction of the vehicle in the frame. Therefore, we had to either put it in front of the shock tower or raise it up high enough that it would see over the shock tower.

Once the camera placement was decided, a mounting tower was designed to sit over the steering linkages and servo that are mounted directly behind the shock tower. Instead of creating a small skinny tower that would rise up behind the shock tower and be potentially unstable, we created a large elevated platform that would create a more stable mounting location for the camera along with some of the other electronics. The large horizontal platform also created a large vertical space to mount the ODROID XU-4, which left the middle section of the chassis available for the batteries and ESC.

The vehicle has a predefined location for one battery, which is typical of radio controlled vehicles, but we had to mount two batteries so a large spike in the drivetrain battery wouldn't interrupt our electronics with a low voltage/ brown out scenario. The decision was made to mount the batteries in a transverse configuration to the original battery mount, so a mounting tab that would attach to the existing battery holder posts near the front of the car was designed. The new tab would support a large band that would go over the two batteries and tuck into the space where the central battery was designed to mount under the rear shock tower. The band would provide enough pressure and friction to the batteries to hold them in place during fast turns and possible vehicle crash scenarios.



(a) 3-D Model ODROID Mount *Tower that mounts ODROID-XU4, Intel RealSense D435i camera, pololu Maestro, and voltage regulator*



(b) 3-D Model of Battery Holder *Friction place battery mount that holds both batteries in place*

Figure 3: Mounting solution for autonomous car

Our mechanical design needed to be fabricated quickly and it went through several iterations, so we decided on using a 3D additive print design. The printing of the parts provided a quick method that required some extra man-hours in the design process to avoid pitfalls of that process such as wide, unsupported spans, but it allowed for less time spent engaged with the actual fabrication method.

Fig. 3 illustrates the CAD models of the 3D printed components that were directly mounted to the chassis. Fig. 3a displays the main shocl tower as seen from the rear passenger side of the vehicle. The vertical surface is the mounting area for the ODROID-XU4 and the smaller elevated platform in the back of the photo was the final position of the Intel RealSense D435i. Fig. 4 is a photo taken in the lab where the transverse mount of the batteries can be seen along with the mounting of the ESC above them on the

battery band.

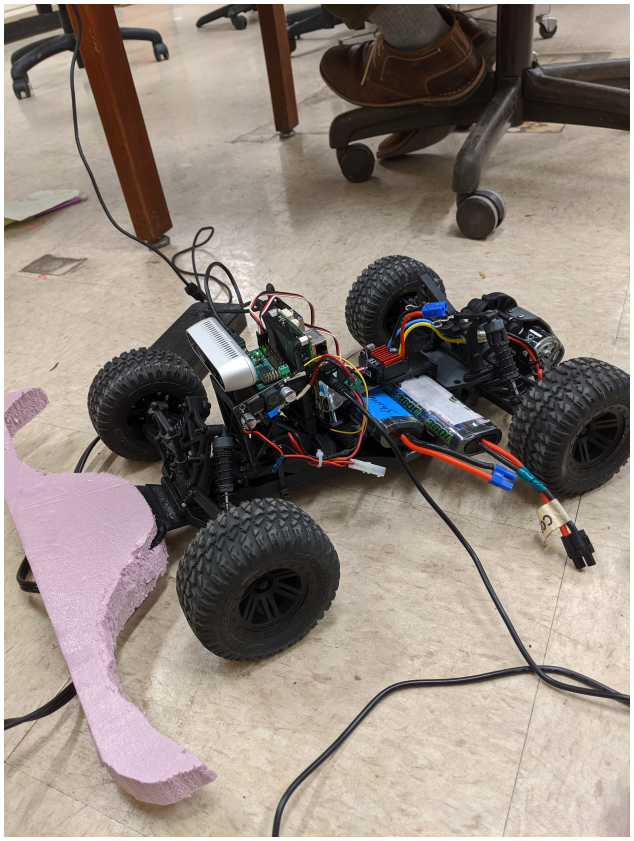


Figure 4: Final as-built photo of vehicle

B. Software

At large, this was a highly software intensive project which required development of many different software types and elements before a final approach was implemented. In general, software development occurred to integrate the RealSense camera, process the image data, and develop a control scheme, which relied on the depth images.

1. RealSense

Integrating the RealSense camera into our system has proven to be a challenging task. A user friendly Robot Operating System (ROS) wrapper exists for the RealSense camera, however, the amount of data the wrapper publishes slows down the computer system and introduces latency. The fix for this issue involved using the RealSense Software Developer Kit directly, leveraging the many capabilities of the API. Our team wrote a roscpp node that extracts depth frames from the camera without publishing all of the RGB data or any of the other sensor data from the RealSense. We also encountered an issue with the format of the depth image. When streaming the values of the extracted image, we realized that each pixel value was represented by a 'lower' and 'upper' value in a format called 'Z16.' This was remedied by converting the extracted RealSense depth image into an openCV matrix using the openCV 'cv_bridge' functionality.

2. State Machine

When a robot executes a complex plan, but all possible states and state transitions can be described explicitly, it can be described with a finite state machine. This approach does not lend itself to general applications for a robotic platform, but it does provide a good framework for task specific robots. Our vehicle has a specific

course and specific tasks to accomplish, so we can define a specific state for each task. We created states for driving down the hallway, turning a corner, and stopping at a stop sign.

3. Computer Vision - Lane Approach

Our team also explored more intelligent ways to extract information from an image using computer vision. In this example we used the Lane Approach.

To follow the corridor is to find the 'lanes' of the corridor. Lane following takes boundary edges from the sides and creates a path in-between them. In our case, the black baseboards on the hallway were the 'lanes'. The black baseboards were converted to 'lanes' by converting a color image into edges.

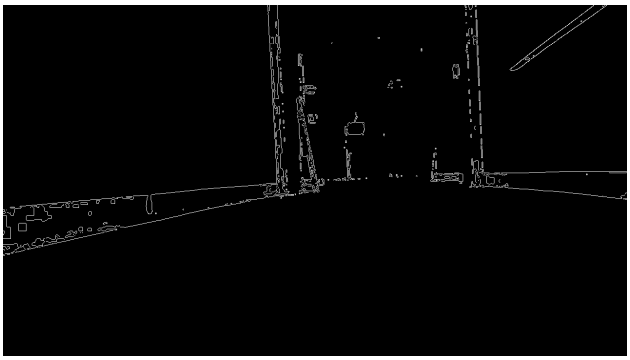
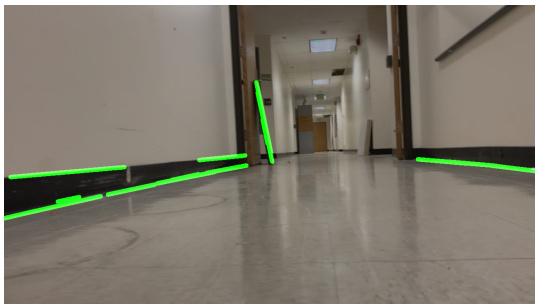
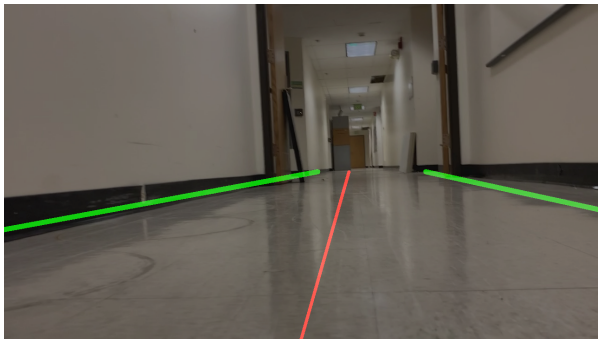


Figure 5: Diagram

The image was then further processed using a Hough Transform. The purpose of the Hough Transform is to perform possible groupings of edge points into object candidates by performing an explicit voting procedure over a set of parameterized image objects. In other words, it finds possible line groupings. The result is shown in figure 6a.



(a) Hough Transform on Edge Detection *Image of finding 'path lanes' on a corridor*



(b) Heading Average *The filtered and averaged Hough Transform*

Figure 6: Illustration of the line following technique

After edge detection had been applied to the image and a Hough Transform was performed, we extrapolated the center line which allowed us to navigate through the corridor.

The disadvantage to this technique was tuning. The edge detection and Hough transform that were used to create the center line to navigate were very unreliable. As a result, we disbanded this method.

4. SLAM

Simultaneous localization and mapping (SLAM) is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it.[11]

A popular SLAM technique is RGB-D SLAM. It allows 3D colored models and shapes to be placed in a point-cloud format. Visual featuring techniques such as SIFT and SURF are used to match pairs of images which are used to measure 3D transformation. The end result is a graph of camera poses nodes and 3D

transformation edges. The advantage of point cloud mapping is seen from creating reach graphs that enable loop-closure and correction. The disadvantage to this approach is that incremental tracking and mapping is prone to drift while requiring a large amount of compute power and time. This approach is available in ROS using the 'rtabmap_ros' package, however, we did not have enough time to pursue it.

Another SLAM method we explored was Orb-slam. Orb-slam uses the same features for all tasks: tracking, mapping, re-localisation and loop closing. The system makes use of ORB features which allow real-time performance without Graphical Processing Units(GPU), providing good invariance to changes in viewpoint and illumination. Therefore it makes the system more efficient, simple and reliable for our use case. However we did not use it because for reason mentioned in the discussion.

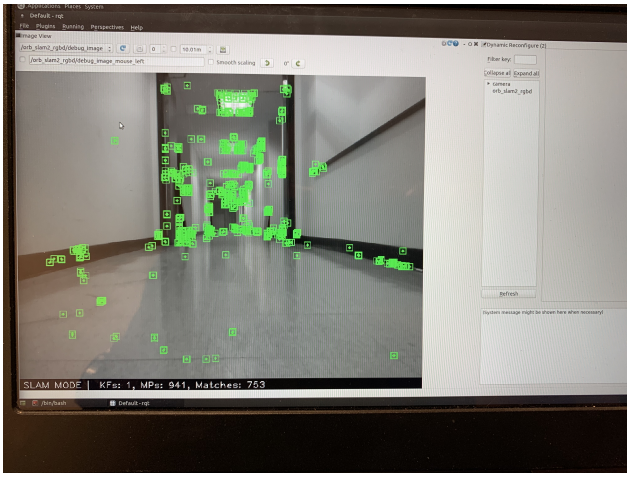
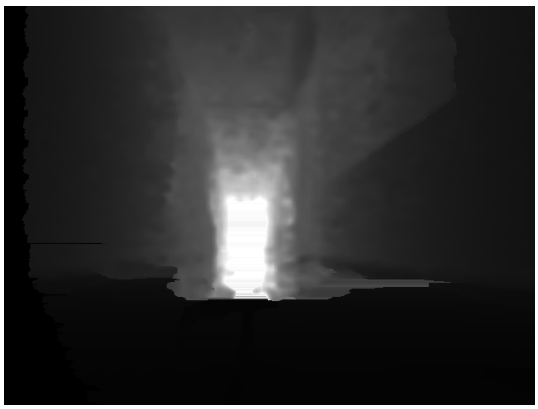


Figure 7: Screen Capture of ORB-Slam object detection

5. Computer Vision - Bounding Box

We implemented a 'follow the light' method that uses depth images. A depth image is encoded as an unsigned-integer type. Figure 8a depicts a visual interpretation of the depth image where white pixels are large values and black pixels represent small values. To be able to process the image effectively and efficiently, the image was blurred and decimated to reduce image complexity. After filtering, a binary filter was then applied with respect to distance to determine open spaces. Fig.8b

The 'Follow the Light' method proved to be very versatile and reliable for long stretches of corridors because the centroid of the object is calculated for the thresholded image within each frame. That said, the threshold value was an important factor in having reliable centroids.



(a) Depth Image *raw depth image from RealSense*

(b) Second Stage *Binary filter added for values of 0 or 255*

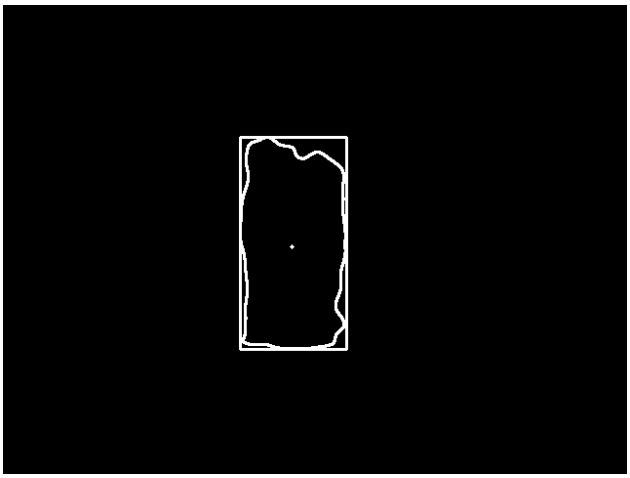


Figure 8: Bounding box and centroid

III. Results

A. Timing

Our robotic vehicle completed the course with an average time of 18.525 seconds, resulting in a second place finish.

B. Challenges

Our vehicle did not successfully complete any of the B-type challenges it attempted. In addition to stopping at a stop-sign and creating a sparse map, the team also attempted the A-type challenge of avoiding a moving obstacle.

IV. Discussion

There are many reasons that the robotic vehicle performed so well in the timing section of the project and so poorly in the challenge section. Primarily, the team prioritized speed navigation of the course above challenges due to the competitive nature of this aspect of the project. Additionally, the team encountered additional problems on the final race day when it was discovered that the vehicle was tuned for the wrong start position. The new tuning for the speed test prevented fine tuning of our challenge code. We discovered that the starting location of the raceway was approx. 1 meter to the rear from where we had anticipated and calibrated for, which introduced a number of errors into the system. This new starting location made it such that the RealSense's depth measurements immediately sent the vehicle into the turning state when it should have been navigating straight down the hallway. The additional runway down the hallway for the robot increased our speed before the first turn, which changed the way the vehicle handled the middle and finish of the turning sequence. The combination of these unforeseen complications resulted in the complete consolidation of effort in the final stretch of the project to focus on calibrating a previously completed aspect of the system. These complications didn't directly impact our pre-existing challenge code, but it cost us critical time that was required to calibrate the challenge code.

V. Conclusion

This project aimed to familiarize students with hardware and software. This was accomplished by having the students assemble an RC car chassis with a Real-Sense d435i and Odroid XU4 for compute. Our most difficult hardware challenge were being able to mount everything on the chassis, but we were able to 3D print a mount that held everything securely in place. Software was then needed to make the car go autonomously around a race course and complete various challenges. Our team was particularly pressed for time so we

tried to optimize the code for race course performance. By using OpenCV to construct a bounding box around the deepest pixels we were then able to use the centroid of that shape in order to output a heading to our servos in order to go to that location. This allowed us to complete the racing portion of the test in just a little bit over 18 seconds. As a group we were very pleased with this time. Overall, we learned a lot over the course of this project and explored what actually goes into making a car platform autonomous.

References

[1]Lecture Notes in Computer Science LNCS: Springer. (n.d.). Retrieved from http://www.springer.com/lncs.

[2]Learning in Robotics, https://pdfs.semanticscholar.org/3ab5/1a737eb363d6b299b6c220067a69999fc845.pdf

[3]Machine Learning Algorithms in Autonomous Driving. (2018, March 15). Retrieved from https://iiot-world.com/machine-learning/machine-learning-algorithms-in-autonomous-driving/.

[4]Robotics Online Marketing Team. (n.d.). Applying Artificial Intelligence and Machine Learning in Robotics. Retrieved from https://www.robotics.org/blog-article.cfm/Applying-Artificial-Intelligence-and-Machine-Learning-in-Robotics/103.

[5]Rosten, E., Drummond, T. (2006). Machine Learning for High-Speed Corner Detection. Computer Vision – ECCV 2006 Lecture Notes in Computer Science, 430–443. doi: 10.1007/11744023_34

[6]Tateno, K., Tombari, F., Laina, I., Navab, N. (2017). CNN-SLAM: Real-Time Dense Monocular SLAM with Learned Depth Prediction. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi: 10.1109/cvpr.2017.695

[7]Zhang, T., Mccarthy, Z., Jow, O., Lee, D., Chen, X., Goldberg, K., Abbeel, P. (2018). Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation. 2018 IEEE International Conference on Robotics and Automation (ICRA). doi: 10.1109/icra.2018.8461249

[8]Luke Burks, Ian Loefgren, Luke Barbier, Jeremy Muesing, Jamison McGinley, Sousheel Vunnam, Nisar Ahmed, 2018, Closed-loop Bayesian Semantic Data Fusion for Collaborative Human-Autonomy Target Search, Final version accepted and submitted to 2018 FUSION Conference (Cambridge, UK, July 2018)

[9]Billard, A., Grollman, D. (n.d.). Robot learning by demonstration. Retrieved from http://www.scholarpedia.org/article/Robot_learning_by_demonstration.

[10]https://arxiv.org/pdf/1902.06162.pdf

[11]https://en.wikipedia.org/wiki/Multi-agent$_system$

[12]M. Desai, M. Medvedev, M. Vázquez, S. McSheehy, S. Gadea-Omelchenko, C. Bruggeman, A. Steinfeld, and H. Yanco. 2012. Effects of changing reliability on trust of robot systems. In Proceedings of the 2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI). 73–80.

[13]Andrew R. Hutchins, M. L. Cummings, Mark Draper, and Thomas Hughes. 2015. Representing autonomous systems' self-confidence through competency boundaries, Vol. 59. SAGE Publications, 279–283.

[14]Steve McGuire, Michael Furlong, Terrence Fong, Christoffer Heckman, Daniel J. Szafir, Simon Julier, and Nisar Ahmed, "Everybody Needs Somebody Sometimes: Validation of Adaptive Recovery in Robotic Space Operations," IEEE Robotics and Automation Letters, Vol. 4, No. 2, pp.1216-1223, April 2019

[15]Steve McGuire, Michael Walker, Jamison McGinley, Nisar Ahmed, Daniel Szafir and Torin Clark, 2018, TRAADRE: TRust in Autonomous ADvisors for Robotic Exploration at the Robotics: Science and Systems, In Robotics: Science and Systems, Carnegie Mellon University (Pittsburgh, Pennsylvania – June 26 - 30, 2018).

[16]Mary T Dzindolet, Scott A Peterson, Regina A Pomranky, Linda G Pierce, and Hall P Beck. The role of trust in automation reliance. International journal of human-computer studies, 58(6):697–718, 2003.

VI. Appendix

A. Report on Human-Robot Interaction with autonomous vehicles

B. Introduction

Human-Robot Interaction (HRI) is a multidisciplinary field of study dealing with robotic systems for direct use by, or in cooperation with, humans. HRI includes, Natural Language Processing, Planning, Target-Search, etc. fields.

C. Discussion

HRI with autonomous vehicles is a many faceted problem with ongoing research to solve many different problems. Fundamentally, it stems from the notion that both the human and the robot have valuable input that can be used to collaboratively solve highly complex problems. One such problem is rogue target-search where an autonomous vehicle is searching an area for a potentially mobile entity. An HRI approach to solve this problem is via the use of semantic data fusion[8] wherein a human can provide data to a continuous, partially observable Markov decision process (CPOMDP) to adjust its search planning. Human input allows the autonomous vehicle to act according to a more relevant reward function and find the rogue target more quickly.

Human trust is an integral aspect of HRI as it directly impacts a human's willingness to interact with autonomous systems. Issues related to human trust in autonomous systems have been explored in a number of independent studies to identify influential factors and their role in human-robot collaboration and supervisory interaction. For example, researchers from the University of Massachusetts Lowell and Carnegie Mellon University modeled changes in operator trust in response to changes in system reliability.[12] TRAADRE (Trust in autonomous Advisors for Robotic Exploration), an investigation into how an operator's trust in a navigational aid can be biased based on the perceived source of aid.[15] Differences in an operator's willingness to trust and follow navigational advice from either a human ground controller or autonomous system were investigated using a series of navigation goals. Operators piloted a moon rover in a Unity graphics scenario with continuous navigational aid from their randomly chosen advisor, where their implicit trust in the aid was determined by the cosine similarity of the rover's current heading to the recommended heading. These investigations serve to quantify human trust in autonomy, however they fall short of analyzing potentially useful information that can be provided by the autonomous system. Research in this direction can serve to improve collaborative efforts as information from both members present in the collaborative effort is considered.

Additionally, improving the effectiveness of human-robot teams can be accomplished by appropriately utilizing information feedback from the system. For example, reference[14] proposes the use of an autonomous system that chooses a human assistant from an available and possibly time-varying pool of human teammates (e.g. astronaut on EVA, operator in a habitat) based on known information about each potential mode of task failure so as to choose the human teammate that is potentially the most effective. The use of the autonomy allows the efficacy of the collaboration to be considered and optimized before a specific operator is even involved. Researchers from the Humans and Autonomy Laboratory (HAL) at Duke University have proposed the use of a Trust Annunciator Panel (TAP) to report the autonomous system's self-confidence as part of a feedback loop back to the human operator.[13] This research provides a basis for machine self-confidence, a metric or series of metrics that describe the system's competency, however HAL did not conduct a human factors study to validate this concept and the TAP isn't necessarily the best way to portray self-confidence information for certain types of space exploration tasks. More recently, researchers at CU Boulder examined the utility of machine self-confidence in its simplest form of implementation. For an autonomous package delivery problem where given a map, starting location, and goal in which operators much choose between 'go' (attempt package delivery) or 'no go' (abort delivery) options for an autonomous robotic vehicle that must navigate between start and goal locations on a map featuring hostile 'bandits' (moving hazards).[16] To assist the operators in making decisions, the autonomy provides two factorized machine self-confidence factors. It was found that providing these factors significantly improved operators' ability to correctly choose between the 'go' or 'no go' options as well as the operators' ability to deploy the autonomous vehicle within its actual competency boundaries. However, this work only serves to validate machine self-confidence in this simple case, ideally human-robot collaborative teams are used to tackle much more complicated tasks with many more degrees of freedom.

D. Report on Deep Learning in Robotics

Deep learning has introduced new applications in the field of Robotics. This section illustrates the various applications, advantages and disadvantages of deep learning regarding robotic systems, particular to a car-like platform.

E. Introduction

Deep learning is of a broader family of machine learning techniques based on artificial neural networks. Deep Learning uses multi-layer feature extraction filters to identify relevant patterns. The shear ability to model complex nonlinear dynamical systems gives deep learning a succinct ability to be used in robotic systems. This field of research where machine learning and robotics intersect is coined as Robot Learning.

Learning has become a major part in Robotics research. Researchers are exploring many areas where Deep learning is applicable. The applications machine learning has influenced are: Robot vision, Robot navigation, field Robotics, humanoid Robotics, legged locomotion, off-road rough-terrain mobile Robot navigation, modeling vehicle dynamics, medical and surgery Robotics.

Furthermore, various learning methods will be central to autonomous systems, as the real world contains too much variation for a robot to expect to have an accurate model of its environment, the objects in it, or the skills required to manipulate them. Using various learning models in tandem can mitigate risks and augment performance in robotic systems. A number of the most notable machine learning technologies utilized in Robotics realm includes; reinforcement learning , supervised learning, self- supervised learning, multi-agent learning , autonomous science, machine learning techniques for big data, imitation learning techniques, Robot programming by demonstration, and multi-agent learning.

One of the main tasks of any machine learning algorithm, e.g the self-driving car , is a continuous rendering of the surrounding environment and the prediction of possible changes to those surroundings. These tasks are mainly divided into four sub-tasks:

- Computer Vision
- Imitation Learning
- Self-Supervised Learning
- Multi-Agent Learning

Deep Learning algorithms are very prominent in four field: Computer Vision, Imitation Learning, Self-Supervised Learning and Multi-Agent Learning.

F. Computer Vision

Computer vision is concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images.In particular, computer vision in robotics is concerned with task such as tracking, SLAM (simultaneous localisation and mapping), localisation, image matching and recognition.[5]

Traditional methods are now being surpassed by deep learning methods. For example CNN-SLAM[6] In particular, the use of deep Convolutional Neural Networks (CNNs) in an end-to-end fashion has demonstrated the potential of regressing depth maps at a relatively high resolution and with a good absolute accuracy even under the absence of monocular cues (texture, repetitive patterns) to drive the depth estimation task.[7]

G. Imitation Learning

Imitation learning is a class of methods for acquiring skills by observing demonstrations. It has been applied successfully to a wide range of domains in robotics, for example to autonomous driving, autonomous helicopter flight, gesturing, and manipulation.

Principle The main principle of imitation robot is that end-users can teach robots new tasks without programming. In a traditional programming scenario, a human programmer would have to reason in advance and code a robot controller that is capable of responding to any situation the robot may face, no matter how unlikely. This process may involve breaking down the task into 100s of different steps, and thoroughly

testing each step. If errors or new circumstances arise after the robot is deployed, the entire costly process may need to be repeated, and the robot recalled or taken out of service while it is fixed.

In contrast, imitation allows the end-user to 'program' the robot simply by showing it how to perform the task.[9]

H. Self-Supervised Learning

Large-scale labeled data are generally required to train deep neural networks in order to obtain better performance in visual feature learning from images or videos for computer vision applications. To avoid extensive cost of collecting and annotating large-scale datasets, as a subset of unsupervised learning methods, self-supervised learning methods are proposed to learn general image and video features from large-scale unlabeled data without using any human-annotated labels.[10]

Autonomous driving and Robotics benefit from this approach because foreign object or new environments not previously labeled may still be properly identified and classified.

I. Multi-Agent Learning

A multi-agent system (MAS or "self-organized system") is a computerized system composed of multiple interacting intelligent agents[citation needed]. Multi-agent systems can solve problems that are difficult or impossible for an individual agent or a monolithic system to solve.[citation needed] Intelligence may include methodic, functional, procedural approaches, algorithmic search or reinforcement learning.[11]

J. Conclusion

Deep learning is technique promising in the field of robotics that traditional methods were unable to deliver. In conclusion, deep learning will play a important part in robotics.