January 28, 2024

The DeepBozo Report

Youwen Wu **Ananth Venkatesh**

Team 1280 Robotics

EECS | Department of Artificial Intelligence

Abstract

DeepBozo is designed to incorporate various AI-enhanced driver assistance features and autonomous technologies. In order to meet our 1 month deadline for a prototype that's ready to scrimmage and train with, Team 1280 EECS needs to conduct a thorough analysis of the goals and limitations of DeepBozo and its various subroutines. In this report, we explore the various possible capabilities of DeepBozo and the feasibility of each.

1 Introduction

We will conduct a holistic analysis of the feasibility of various DeepBozo features, in order to operate as a utility-maximizing subteam and utilize our resources at allocative efficiency, while developing our planned features within the limitations that are imposed upon us.

1.1 Methodology

We will discuss each planned DeepBozo subroutine and classify these planned features on 1 of 4 levels: trivial, feasible, insane, and ludicrous.

- **Trivial**: this means that the feature should be straightforward to implement with little advanced knowledge or research required and likely contains a large amount of "grunt work." This task can probably be offloaded to rookies without much issue.
- **Feasible**: this means that the feature is most likely doable with a bit of elbow grease and some digging around on StackOverflow. Any member of programming can probably complete it without much issue, but it may require more in-depth codebase knowledge or advanced skillset that rookies may not be able to meet without some assistance.
- **Insane**: these features are difficult to implement. It's not certain whether they can be implemented fully according to their specifications, and they will likely require quite a bit of work, experimentation, debugging, and reading. These tasks likely require a somewhat advanced programming skillset and an in-depth understanding of the codebase and robot. It's probably not a good idea to assign these to rookies.
- **Ludicrous**: these features err on the side of science fiction. It's not certain whether even a rudimentary proof-of-concept can be created for these features that works well enough for testing, let alone a production-ready release developed up to the original specifications. It's probably best not to attempt too many of these features before completing a few less challenging ones.

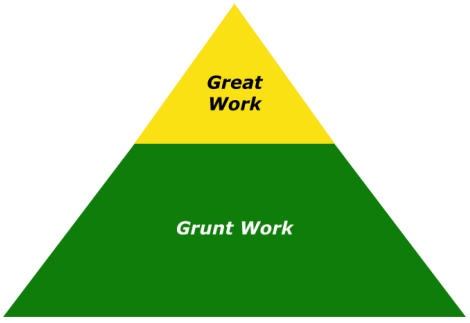


Fig. 1: Grunt work (which is common in **trivial** and some **feasible** tasks) lays the foundation for great work. Offloading this to rookies maximizes marginal revenue product, allowing us to operate at allocative efficiency.

1.2 DeepBozo Nomenclature

The DeepBozo autonomous suite's various subroutines have been classified into a few distinct groups based on their general specifications, due to its complex nature (see **Fig. 3**). However, they are all designed to work in tandem, so the various subroutines may rely on and utilize each other's features. The subroutines are defined as follows:

- **BozoVision** (GAVIN, "General Autonomous Vision Information Networking"): our computer vision suite. This mainly encompasses features involving object detection, classification, and recognition via Limelight, accelerated by the Coral TPU and other hardware at our disposal (e.g. the ROG Zephyrus). It also includes other sensing capabilities like the LiDAR. This is currently the most mature subroutine thanks to the extensive work done on the Limelight so far, but artificial intelligence integration is not fully complete.
- **BozoAssist** (AMIT, "Automated Movement In Teleoperated"): our driver assistance capabilities. These features mainly pull data from the other subroutines and synthesize them to make real-time decisions to assist the operator and avoid catastrophic incidents. BozoAssist may periodically override teleoperated commands when deemed necessary or when operator performance is suboptimal. This subroutine relies on, but does *not* encompass FSD (see **BozoAuto**).
- **BozoLLM** (LMAO, "Large-language Model Autonomous Operations"): our large language model, with potential vision input, running on the driver station laptop to bypass coprocessor restrictions. This will be the decision-making subroutine which will assist the autonomous mode and the operator with its intelligent capabilities, when our human-made algorithms fall flat.
- **BozoAutonomous (Mk. II)** (FSD, "Full Self Driving"): our flagship autonomous mode. It will utilize a pre-planned path for its initial autonomous routine, but feature automatic decision making with the capability to drive the robot and complete objectives without any human or operator input. This subsystem may also be activated during the teleoperated period, especially recommended when driver performance is suboptimal (as determined by **BozoAssist**).

Fig. 2: Elon Musk brilliantly states the guiding philosophy of the **DeepBozo** project, which is to replace incompetent and error-prone human operators with highly-trained and specialized intelligent subroutines.

2 Diagrams

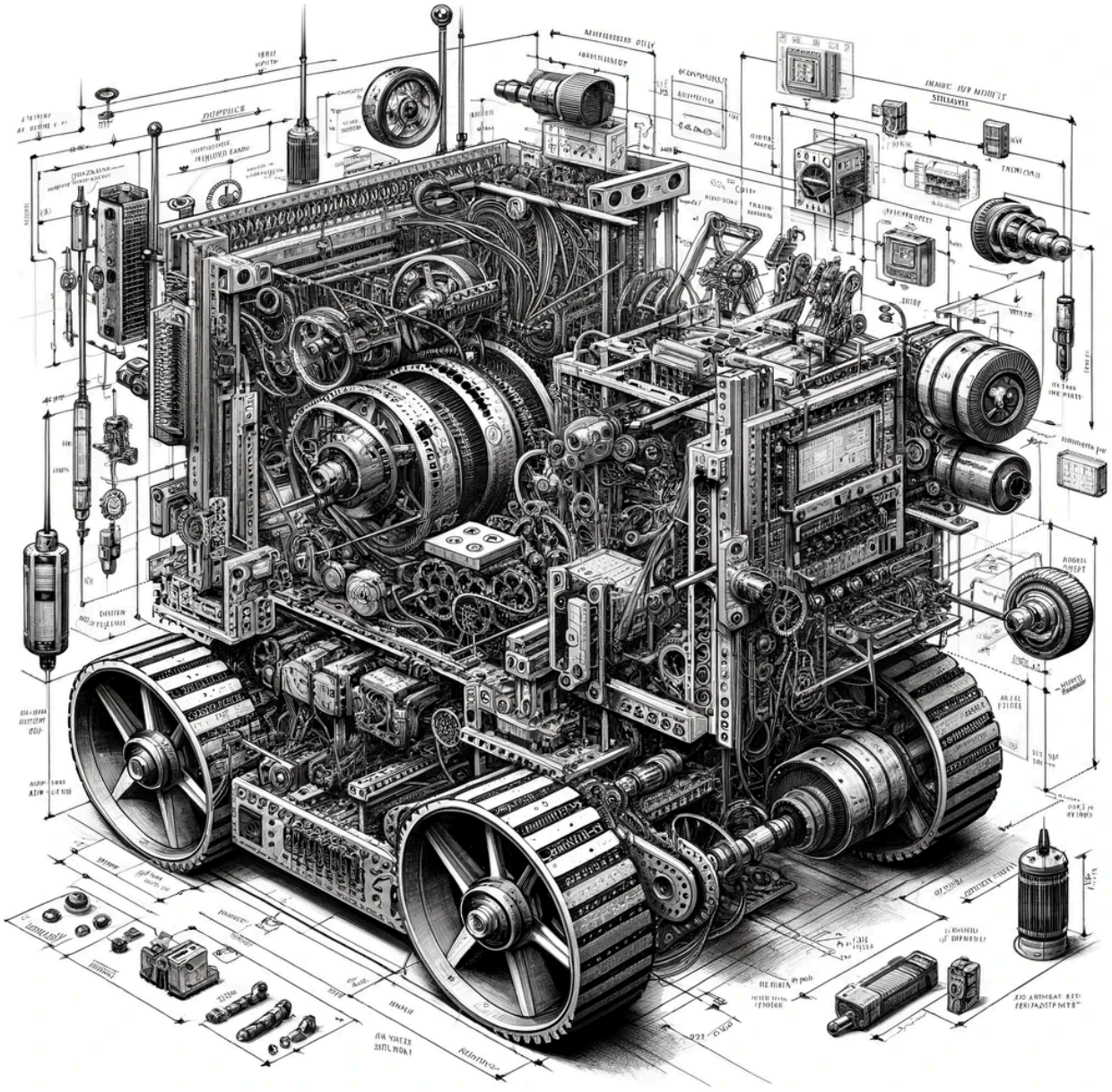


Fig. 3: DeepBozo internals, visualized

3 Planned Features

We currently have various planned subroutines that we need to implement in the next month. However, we are lacking a concrete plan as to which specific features actually need to be created. In this section, we will analyze the various submodules that the DeepBozo subroutines each consist of and their feasibility.

Note: some of these features are already implemented. They are included for posterity.

3.1 BozoVision

Feature	**Description**	**Feasibility**
Rudimentary Vision	Allow the Limelight to automatically detect AprilTags and other visual fiducials. This is the baseline level of vision.	**Trivial**: this is technology built-in to the Limelight
Object Recognition	Use the Limelight to classify basic objects and, more importantly, detect game pieces and other commonly-encountered objects.	**Feasible**: this is functionality that the Limelight is designed to do, but it needs the Coral TPU.
Occupancy Network	Use all vision systems to generate an approximate object depth map. Odometry should allow the robot to know where it is on the field, and the LiDAR can ground depth estimations by calculating the distance to a specific location on one of the Limelight images (with a rotating mount, this can potentially be used to generate a linear depth map). Ideally, a visualization would also be incorporated in the Jankboard. To expand on the LiDAR's capabilities and realize the full potential of the Occupancy Network, a depth model should run on the driver station laptop in real-time.	**Ludicrous**: this is functionality we will have to develop almost entirely from scratch, incorporating Limelight and LiDAR information. The depth model, if running concurrently with **BozoLLM**, will face certain hardware limitations. Notwithstanding the depth model, this feature may be classified as **Insane** instead.

3.2 BozoAssist

Feature	Description	Feasibility
Aimbot	Automatically align the robot to a specificed target (likely the game pieces).	**Trivial**: following April Tags with **BozoVision** will allow a trivial implementation of this with game targets.
Cruise Control	The robot attempts to reach and maintain a set speed, adjustable by the operator.	**Trivial**: simple to implement, nothing fancy here.
Adaptive Cruise Control	Includes all the features of regular Cruise Control, but the robot also dynamically adjusts speed smoothly to avoid forward collision.	**Feasible**: Once LiDAR is properly mounted, it shouldn't be too difficult to add this functionality onto existing Cruise Control.
Automatic Collision Mitigation	The robot automatically avoids predicted imminent collisions using the available sensors, or if implemented, the **BozoVision** Occupancy Network.	**Feasible/Insane**: depending on the capabilities of the rotating LiDAR, this feature may be difficult to implement. Relying solely on vision seems technically difficult, without the depth network. If the scope of this feature was reduced to just forward collision mitigation with a front-facing LiDAR, it may be instead classified **Trivial/Feasible**.

3.3 BozoLLM

Feature	Description	Feasibility
LMAO – LLM tuned for robot control	An LLM that is able to provide directions that can be converted to instructions that control the robot. Optionally, this LLM can include image recognition (using e.g. Llava).	**Feasible**: the main issue is figuring out how to integrate the LLM with the robot controls and telemetry.

3.4 BozoAutonomous Mk. II

Feature	Description	Feasibility
Pre-planned paths	Planned paths for the autonomous mode.	**Feasible**: shouldn't require anything fancy, but rookies cannot be trusted with it due to its critical strategic importance.
FSD + Intelligent path-planning	Intelligent path planning via DeepBozo subroutines when the pre-planned path is disrupted. The robot should be able to drive itself and automatically avoid obstacles in its path, using both **BozoVision** and **BozoLLM** output to guide its path.	**Ludicrous**: Depends on how trustworthy we need the FSD system to be and whether it can be used in the teleoperated period.

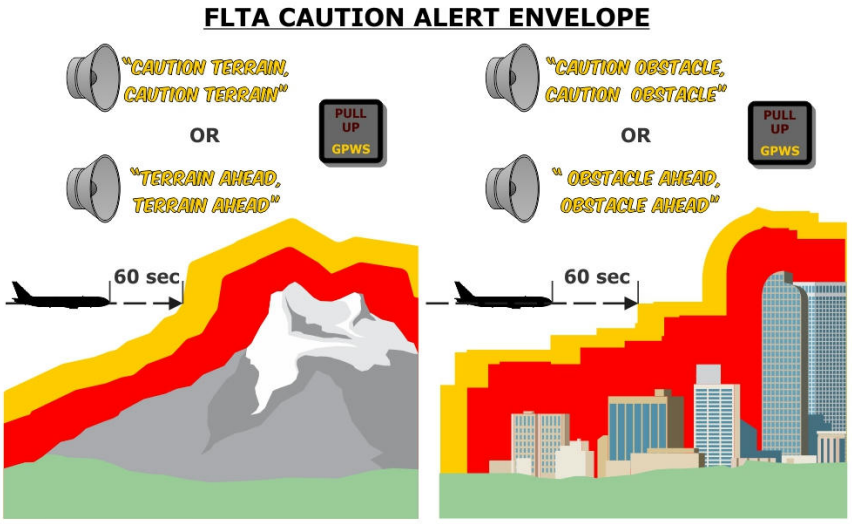
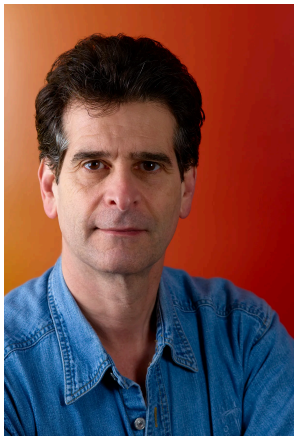


Fig. 4: Just as the now commonplace GPWS system regularly prevents 9/11 incidents, **DeepBozo**'s incredible breadth of features will correctly guide human operators even during unexpected circumstances due to its intelligent decision making capabilities.

4 Conclusion

Clearly, the development of DeepBozo will be a task filled with ardor, tumult, and obloquy. We can see that some DeepBozo features may be somewhat unrealistic. We should add and assign the various features in each subroutine to people on GitHub, and focus on **trivial**/**feasible** features for now so that we have basic features ready for testing and scrimming.

Pictured above: The ROG Zephyrus (*left*), a critical hardware upgrade that will allow running many **DeepBozo** subroutines. Dean Kamen (*right*), an FRC hero who also likes to visit Epstein island.