Indy Autonomous Challenge

Kennesaw State University

Round 1 Whitepaper

Team Composition

The Indy Autonomous Challenge (IAC) team from Kennesaw State University (KSU) is led by a team of faculty members from various Department in the University:

- Kevin McFall, Ph.D. – Department of Mechatronics Engineering
- D. Michael Franklin, Ph.D. – Department of Software Engineering and Game Development
- Bill Diong, Ph.D. – Department of Electrical Engineering

These faculty members all have experience working in autonomous systems with Dr. McFall[1] leading numerous undergraduate student teams in autonomous vehicle projects[2] over the past 5 years, including in full-size automobiles. His research has led to publications using image processing to locate and control an autonomous vehicle[3]. Dr. Diong[4] also works directly in autonomous vehicles, having been funded by the Georgia Department of Transportation to develop a semi-autonomous bus rapid transit vehicle[5]. These engineers are joined by computer scientist Michael Franklin[6], whose work in building AI frameworks for multi-agent autonomous systems[7] is key to developing the computing architecture for this project.

KSU has several student competition teams, including the Electric Vehicle Team and the Autonomous Ground Vehicle Team, from which to recruit talent. Undergraduates from these teams will be joined by graduate students in Computer Science and Applied Engineering – Electrical. The basic skills necessary for the IAC are already being acquired by these students as their yearly competitions require building similar software stacks to that required in the IAC.

Existing Autonomous System

The KSU solution for the IAC will build upon experiences from entry in the Electric Vehicle Grand Prix autonomous series. The autonomous go-kart entered in that competition includes components for perception, navigation, and control. This is all on top of a fail-safe safety system, of which is on an embedded system separate from the main autonomous system.

The perception stack is composed of a standard occupancy grid SLAM implementation, as the competition rules allows for pre-mapping of the track. For perception and tracking of other karts while racing wheel-to-wheel, April tags are used for accurate tracking and estimation. On top of this perception stack, the system is also using an RTK-GPS for accurate localization within the track.

[1] https://scholar.google.com/citations?user=BXagjy0AAAAJ&hl=en
[2] https://www.youtube.com/playlist?list=PLiAXye54pUSYUB9oqlD2BQqJNKFU4OkdJ
[3] http://doi.org/10.1007/978-3-319-33681-7_77
[4] https://scholar.google.com/citations?hl=en&user=_TME_VMAAAAJ
[5] https://doi.org/10.1109/SECON.2017.7925292
[6] https://scholar.google.com/citations?hl=en&user=bJbypRUAAAAJ
[7] http://doi.org/10.4018/IJMSTR.2017010101

The navigation system is built similar to ROS' navigation, with a global and local planner for pathing. The global planner identifies the path around the entire track at once, minimizing curvature. The local planner determines how to follow the global path, and is tied into the controls of the kart.

Model predictive control uses the state of the kart as well as the state of the other karts around it to influence the local planner, the goal of which is to follow the global planner as closely as possible to have the fastest lap possible.

Extending Beyond ROS

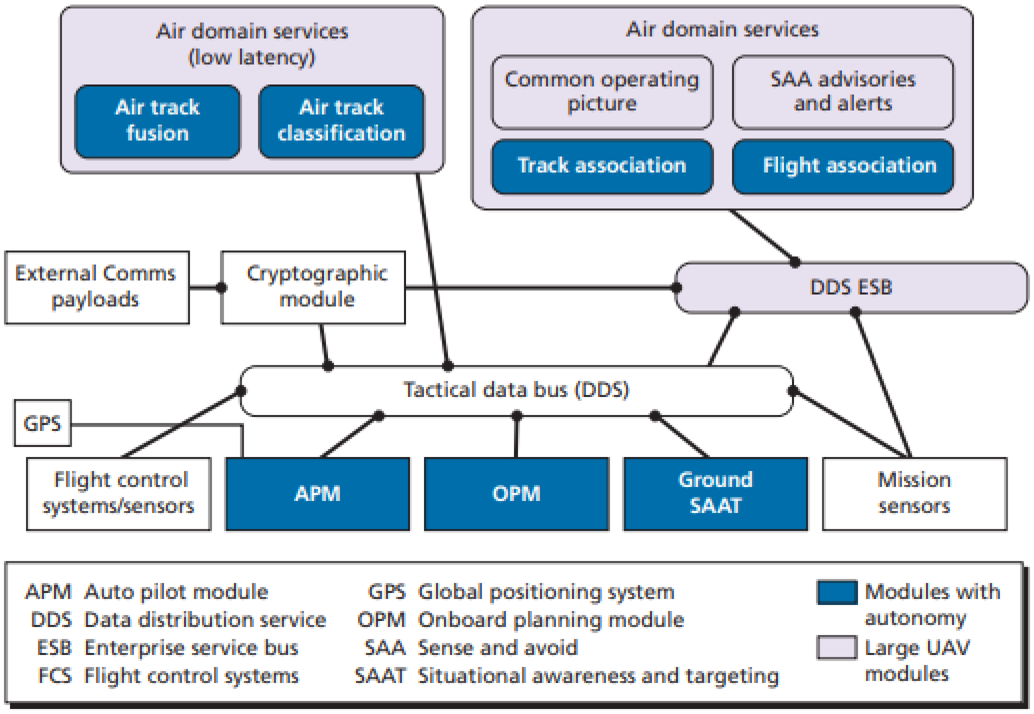
While there are several extant systems available for specific type of robotic interaction and autonomous control, such as ROS, we believe that we will need two distinct operational capabilities that may prove difficult these existing systems and schemas. First, we wish to operate with very high response rate due to the challenge presented in this task (namely, high-speed, real time image recognition, navigation, obstacle avoidance, control, and safety). ROS, as an example, uses many serial segments and round-robin polling techniques without true interrupt ability. While this is fine for slower robotics and control applications, we are certain that speed will play a crucial role in the success of this project. Second, we believe that a distributed and parallel autonomous control system (ACS) will offer increased performance, greater safety, and strong perception of the environment. We wish to have a hierarchically arranged distributed intelligence where each element of the ACS has its own ability to make decisions independently, but still work according to the distributed plan from above. This distributed autonomy is mission critical to high-speed navigation and safety.

To this end, we propose a system built on distributed computing cores with custom-built software utilizing high-functioning libraries. This may vary from Python libraries running on Raspberry Pi's to TensorFlow running on NVIDIA Jetson Jetpacks. These distributed machines will all be redundant so that there is a local backup and distributed so that they have hierarchical backup. Each 'leaf node' of our distributed ACS will be a set of intelligent computational devices, as mentioned, that will handle local calculations and processing of their core task while communicating task-oriented information upstream to the higher levels of the hierarchy. In addition, the highest level of the hierarchy, likely to be a well-qualified PC, will have the ability to remove sub-elements (the leaf nodes mentioned above) and take over their function in cases of malfunction or damage. There will also be times when the top-level node (the PC) will be making its own decisions to compare with the leaf nodes to ensure maximum or optimal performance. Overall, this top-level node will be responsible for determining the current mode of operation (e.g., stop, crawl, pace, race) and current best strategy for achieving the desired results. It will then send this strategy down to each level so that those connected nodes can switch their policies to match this current strategy. This is a core principal for our ACS, distributed, but connected, intelligence.

We will attempt to leverage an existing partnership with outside companies like NVIDIA to help create these custom solutions. The Jetson Jetpack has a solid library of high-powered GPU-based AI tools, like computer vision and TensorRT, that will help with on-board processing. These elements are also used in their autonomous vehicle system and they are constantly innovating within this space. The NVIDIA devices also inherently run CUDA, so our custom code will also enjoy the speedup from this technology. Additionally, we will explore open-source solutions and smaller-scale solutions at each node to maximize performance, reduce costs, and offer heightened redundancy. In either case, we believe this will help us to realize our goals of increased speed and distributed intelligence.

Proposed Autonomous Architecture

The Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics sponsored research to advise the military on an architecture standard for military various autonomous systems[8]. The architecture the KSU will adopt is adapted from the conclusions of that research, whose system architecture for unmanned aerial vehicles appears below.



Source: Designing Unmanned Systems with Greater Autonomy, Daniel Gonzales and Sarah Harting

Without the need to store mission data, all communication will require low latency and be handled on a single data distribution service (DDS). The primary system components and their requisite sub-functions are:

- Onboard sensors
  - Individual modules to acquire data from each sensor and publish to DDS
- Localization module (not appearing in diagram)
  - Use Monte Carlo, particle filter, or similar localization technique to identify global vehicle pose using sensor data and given map of track
- Track fusion (classification will be unnecessary in a closed track)
  - Individual object identification algorithms for each sensor (radar, LiDAR, vision, etc.)
  - Sensor fusion and tracking using unscented Kalman filter or similar technique
- Sense and avoid (SAA) alerts
  - Prediction for likely collisions given tracking results
  - Decision structure to override APM when warranted by collision prediction
- Onboard planning module (OPM)

[8] https://www.rand.org/pubs/research_reports/RR626.html

    - Determine desired global path for current lap and encode it as a set of waypoints
    - Identify the next waypoint as the current target destination
- Auto pilot module (APM)
    - Convert waypoint destination and global pose into desire velocity twist for the vehicle
- Control system
    - Convert desired twist into actuation commands for steering, acceleration, and braking

Summary

The KSU IAC is formed with experienced faculty and motivated students. The team brings with it experience programming and deploying autonomous systems, and is well positioned to develop a viable entry in IAC. The team is currently seeking sponsorships from local industry partners and hopes to secure funding to support it in Round 4.