CSE145
Final Report
Project Tau

Qian YingDong
Cao Yue

Abstract:
Project Tau is intended to build a multi-agent system of single-leg robots that can be arbitrarily assembled to accomplish a common goal, such as walking, running and rotating. Given a novel task or situation, the system explores a satisfying strategy through distributed machine learning. While each component robot independently decides its best next move based on real-time sensor data and its own experience, the movement of the distributed system eventually converges to a global optimum through the collaboration and communication of each robot. The merits of this project lie firstly in the distributed processing to exploit cheap hardware to meet the intensive computing demands, and secondly in the simplicity of the robot's physical structure to allow the system to be scaled and applied without exorbitant costs. This paper focuses on the work done in the first ten weeks of the project: the hardware design and realization, the program flowchart and an exploratory machine learning algorithm.

1 Introduction

Multi-agent systems have always been a primary research field in distributed artificial intelligence(Shoham, Powers, & Grenager, 2007; Weiss, 1999). It resembles certain real-world models where several entities interact to accomplish an agreed goal. Such model ranges from army ants go on a forage for prey(Rubenstein, Cornejo, & Nagpal, 2014), to rescue teams searching for survivors in the ruins. Although these models can be analyzed using centralized models, practical constraints such as the restricted processing power of a single chip, exponentially growing sample dimensions when scaling the system, and consequently sparser sample size, are better dealt with using multi-agent learning models. In general, a multi-agent learning model follows three assumptions (Weiss, 1999). First, a single agent only has local knowledge. This assumption is imposed by the distributed nature of the model. The limited capacity of a single agent's sensor and storage restricts how much it can receive from the environment and how much it can process. Second, a single agent only has limited impact on the whole system. It is the interaction of all agents and interconnection between the decision and action of each agent that this model focuses on. Third, agents can prevent its neighbors from executing certain actions. This assumption has its realistic concern. In a cooperative system when work is done by negotiation and compromise, one agent's decision may conflict with another, and how to resolve this dissension is significant in converging to an optimal solution.

The initiative of Project Tau is to investigate the behavior of a multi-agent system while reducing the model's complexity to a manageable degree. Therefore, the hardware structure and capability of a single robot are minimized. Hence, Project Tau naturally follows the above assumptions. In its tentative final stage, all the agents will be equipped with an IMU (Inertial Measurement Unit) collecting its real-time movement information, a on-board chip reading the IMU data and communicating with its neighbors, and a servo-driven leg capable of moving in one dimension based on the commands sent by the chip. As a single agent can only process the information of how its direct neighbors are going to move next, it only has local knowledge of the whole system, but it still enables the robot to make a decision of the next movement without interfering with others. Moreover, as all the robots are connected to form the whole system, every agent has an impact on the whole system, but has no guarantee of whether the system will move in the way it intended to. The behavior of the whole system is the aggregation of every component, where each component has a limited influence. Finally, it is imaginable and is an observed fact that nearby agents can have contradicting actions. For example, when two agents try to move their legs closer to each other, their legs may collide and are locked from further movement. How to prevent such conflict from happening, and when it happens, how to resolve it, is one of the primary concerns of Project Tau.

In order to pursue the initiatives in an efficient and well-organized manner, this Project is divided into three phases and the first two phases were completed during the Spring quarter of 2016. The first phase mainly focuses on hardware design and simple algorithm implementation to better prepare the system for future phases. Different prototypes were 3D-printed to test its usability and several modifications were made. The second phase addresses itself to the machine learning element in this project, and attempts to modularize the host program to provide solid workspace for testing machine learning algorithms. This report covers the technical materials of the project, its milestones as well as its visions.

2 Hardware Design

This section mainly focuses on the hardware design of the unit robot and the construction of the group robot.

2.1 The Unit Robot

The Unit Robot consists of the mechanical structure and the electronic part.

2.1.1 Mechanical structure

The unit robot consists of three 3D-printed body parts: a head frame holding all the parts together, a servo-driven leg, and a support platform holding all the electronics.

2.1.1.1 Body (Frame)

The first component is the cubic frame connecting all its components together. When designing this cubic frame, we paid close attention to three attributes: stable, light and easy-to-assemble. As the head of the robot, the cubic frame must be able to hold the servo as well as the screws without being stretched out of shape. A cube box with five solid surfaces would be considerably firm, however such design would add excess weight to the system, and difficulty in the assembling process. Therefore we

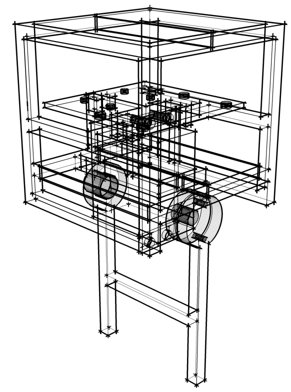

Figure 1: The Overview of the Unit Robot

finalized the design to a lightweight cubic frame with several cross beams to share the load. Support structures were also added to the frame to prevent the servo from oscillating.

2.1.1.2 Leg

The robot leg is of the shape "H" but is not completely symmetric. On one side of the leg, there is a solid disc at the top used to connect the steer of the servo. Two holes will be drilled in order for the nails to pass without damaging the printed structure and strength of the leg. The opposite side has a ring in order to stabilize the leg. The leg is printed in a fashion that guarantees its strength. However, the 3D-printer used is not accurate enough to ensure that every leg is of the same length and its ring is round enough. Extra polishing steps are needed in order for the leg to move smoothly. Such handwork induces noise in the system. In addition, the friction between the leg surface and the ground, which is why the robot can move at all, is too complicated to analyze due to printing inaccuracy. This adds difficulty when we tried to simulate the robot.

2.1.1.3 Support Platform

The last 3D-printed component is the support platform for the electronics. The platform is engineered to hold the IMU and the Wi-Fi chip ESP8266. The platform is attached to the main frame through three points to prevent it from vibrating.

2.1.2 Electronics

Features and choosing standards are discussed in this section.

2.1.2.1 Basic rule

There are a large numbers of microprocessors and sensor chips available in the market. Based on the need of building scalable, low-cost group robots, the weight, size, price, communication means and performance are the main attributes that we took into consideration.

2.1.2.2 ESP8266

ESP8266 is the cheapest WIFI-embedded microprocessors in the market. It features small size, light-weight, high performance in terms of its size. It also supports SPI to communicate with the sensor. Most importantly, it has full TCP/IP stack which can be utilized for communicating among peers and with the host computer. NodeMCU is an embedded system designed for ESP8266, well supported by an open source community. Basically, it abstracts the lower-level code of ESP8266 chip to higher, user-friendly codes. It also provides hardware interface to debug the system as well as communicate with other sensors.

2.1.2.3 BNO055

The IMU used for this project is BNO055, an absolute orientation sensor that deploys sensor fusion algorithm to offer the actual 3D orientation data calculated from an accelerometer, gyroscope and magnetometer. We started with a different IMU called MPU6050, but it was too difficult to work with. We later found out that calculating the absolute orientation from any 9-DOF (Degree of Freedom) sensor is too challenging and time consuming for our purpose. Therefore, we chose to use BNO055, and get the ready-to-use orientation for Project Tau in order to speed up the developing process, despite its high price.

2.1.2.4 Robot Servo

A servomotor is a rotary actuator that allows for precise control on angular position, velocity and acceleration. Different from a simple motor, it is coupled to a sensor for position feedback. In our project we assumed that the servo can go to the required position after a small amount of time. This is because the power of the hobby servo motor is relatively strong compared with the weight of the robot. The servos used in this project are all analog standard servos, capable of generating up to 11.0 kg·cm Torque and rotating 60 ° in 0.2 seconds. This power is enough for the servo to support the robots and drive them to walk. However, servo may not be a suitable solution to much faster locomotion such as jumping or leaping.

2.2 Group robot design

The core of Project Tau lies in the collaboration and communication between all the unit robots. This imposes several requirements on the group robot: first of all, the power supply used for a single robot is no longer capable of supplying a group of robots. Simply multiplying the number of power cables is an undesirable solution due to the weight and constraint caused by all the wires and plugs. Second, how to physically connect these robots together is a tricky question. The connection must tolerate the varying and even contradicting forces generated by movements of robots. It also needs to be flexible and convenient for the assembling and reassembling process. Although the answer to this connection problem is still open, in current stage tape is used as a tentative solution. Finally, as the unit robot was designed with simplicity in mind, the group robot can be assembled in almost any way. How to exploit this structure flexibility and versatility is one of the main focuses of future phases.

2.2.1 Power supply

A single robot is charged through a 5V, 1.2A USB cable. This method soon becomes cumbersome when we shifted the focus of our project to a four-unit group robot. Unless four of these USB cables are used, the voltage is too low to power up the servos. However, the unbalanced weight induced may hold back the group robot and prevent it from moving forward. A clever way to tackle this problem is to use ATX power supply. It is intended to serve as a computer power supply, however, its ability to supply 20A at 5V makes it a perfect power source to drive the micro-controller and the servos.

2.2.2 Connection

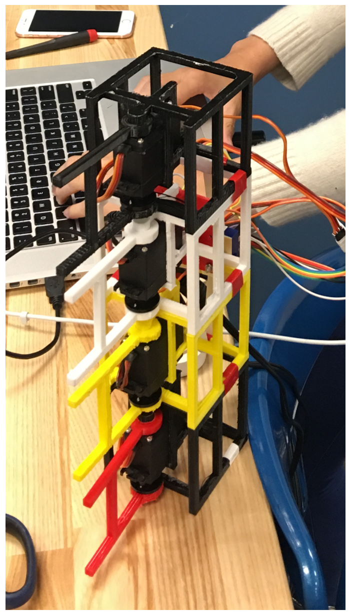

Figure 2: A Sample Connection of 4 Unit Robots by Tapes

How to connect all the unit robots together to form a stable group robot has always been a major concern of our project. In general, such connection must comply with the following two requirements. First, it must be durable and secure. As each unit robot has freedom over its own action, it is possible that in early learning states, two neighbor robots will move in contradicting ways to generate an appreciable internal force which may break the connection. An ideal connection should be able to cope with such situation to provide a stable learning environment for the group robot, so that the system can eventually find a reasonable moving strategy. Moreover, even for an optimized moving strategy, there is still potential risk of shifting and vibration. A stable connection will no doubt reduce such risk.

The other requirement is that an ideal connection must be easy to assemble and reassemble. The essence of Project Tau lies in the fact that each unit robot can be used and reused to form various group robots. Fail to meet such a requirement degrades the flexibility of the system we have in mind.

An elegant solution to the connection problem is still unknown to us. However, as in current phases we only need to work with a few group robots, a simple, interim solution is taping all the unit robots together. Tape do meet the two requirements mentioned above, unfortunately it cannot be applied in a larger scale.

3 Software

This section mainly focuses on the software design of the unit robot and the host computer.

3.1 ESP8266

Our program run on ESP8266 is iterative. First, it fetches the sensor information from BNO055, adds a timestamp and wraps the whole data into an HTTP POST request. If the wireless connection with the host can be successfully established, it sends the POST signal and waits for the host's reply. Upon receiving the reply, the program decodes the moving instructions calculated by the host, and converts them into PWM signals that the servo understands. This routine is repeated about every 0.01 second.

3.1.1 Servo

Servos are controlled through PWM signals. Each servo is calibrated based on its position at 0, 90 and 180 degree. Then a converter program will make sure every servo motor reacts the same to one control signal.

3.1.2 Wi-Fi Connection

We build a private network between the robot and the host to maximize the bandwidth and reduce the latency. A router is directly connected to the host computer. It establishes a wireless network named Tau. When the ESP8266 is initialized, it will first search for the wireless networks and try to connect to it. Once the connection is built, data will continuously be exchanged via this interface. Based on our test, this solution is far better than directly connecting to the host or using the ESP8266 as a wifi hotspot.

3.1.3 HTTP Transmission

We use HTTP protocol to boost the connection between the robots and the host. Instead of rewriting a new communication protocol, we referenced the design of web server maximizing the network capacity. We tested three http protocols in total and found out HTTP 1.1 with reused connection provides relatively fast transmission and saves the resources for both the host and the embedded system. One network transmission process can be done in 0.005s. However, there are a few potential delay problems during the transmission.

3.1.4 Sensor data processing

We first establish the connection between the sensor and the ESP8266 via SPI, and then get the raw data of the sensor from the register. To save the energy of the microprocessor, the useful data is packed into a 30 bytes array and directly sent to the host. This data array consists of gyroscope, linear acceleration, Euler angles, Quaternion and the time stamp.

3.2 Host:

The flow chart illustrated the structure and the information exchanged among different processes and threads. A detailed explanations are in the following subsections.

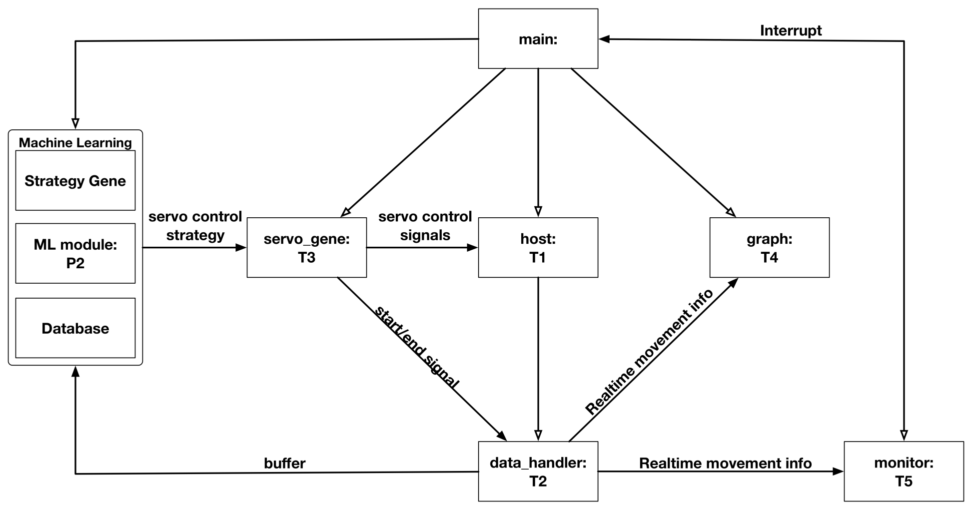

Figure 3: Flow Chart of Host Program

3.2.1 Main

The codes in main.py are responsible for the control of processes and threads of the program. It will start the processes and threads in order and exit the whole program if any process or thread fails.

3.2.2 Graph

Graph module is responsible for visualizing the IMU data and plotting diagram of accelerations, velocities and displacements. We use OpenGL and pygame to draw the IMU, and visualize the row-yaw-pitch data by doing 3D transformation to the virtual object. We also keep a list of trajectories so that the route of the object can be traced back in 3d space. We integrated the acceleration and used Matplotlib to plot the diagram. The diagram refreshes based on the movement of the robot. The diagram benefits us a lot in improving the design of machine learning module.

3.2.3 Servogene

The servo signal generator serves as an intermediate between the machine learning module and the host program. It fetches the movement strategy generated by the machine learning module, and translates the strategy into forms understable to the host program, so that the instruction can be sent to the group robot for execution. This generator also signifies and start and end of an action, so that the data handler can take corresponding actions.

3.2.4 Host

We wrote a HTTP post handler to deal with the incoming data, and pack the latest servo control signals to the reply packet. We did several pressure test on this server, and it can handle the concurrent 2000 connections per second. We can safely assure that the host program will not be the bottleneck of the whole system in this phase.

3.2.5 Datahandler

Data handler is the most versatile part of the program. It parses incoming data into a list, converts the unit Quaternion into Euler angles, verifies the incoming packet, integrates acceleration to get velocity and displacement and sends the valid movement data buffer to the database as well as the graph module. Basically, data handler is the intersection of the data flows. However, in the later version of the program, its tasks will be allocated to several independent modules.

3.2.6 Monitor

The monitor module oversees the movement of the group robot throughout the learning process. It serves as a safeguard program and reacts upon any undesired event. For now, it interrupts the main program whenever it detects an unusual orientation data signifying an overturn of the robot. The host will terminate upon receiving the interrupt, thus preventing the robot from damaging itself.

3.2.7 Machine Learning Module

The machine learning module consists of three major components. Strategy generation module schedule the queue of movement execution. ML module updates the value of learning parameters for this stage. Database handles and stores learning data.

3.2.7.1 Strategygene

Based on the existing movement strategy, strategy generation function will evaluate and continuously generate sequences of control signal for the robots to move. By schedule the order of the strategies, so the ML module can get the intended information according to its priority.

3.2.7.2 ML Module

For this stage ML module is a value updater. Based on the initialized states of the robots, it modifies the parameters that control the robot accordingly. Take standing as an example, P_1 is the position angle of the first servo motor. Let positive feedback $\tau = 1$ and negative feedback $\tau = -1$. Let last time modify angle be m_l. Thus, the simplest value update equation can be written as:

$$P_1 = P_1 + \tau * m_l$$

By changing the value of parameters, the strategygene can be a useful tool to verify its updated parameter. After some time, the parameters will be converged and the optimal solution is found.

3.2.7.3 Database

For current stage we simply use the file system as a temporary storage. All the useful data processed by the data handler will be sent to a separate process responsible for I/O operation. It stores the data into a txt file in the form of JSON. We plan to use non-SQL database in the future to help us store and sort all the valuable information learnt.

4 Milestones

Followed is the milestones set at the beginning of the quarter, while changes made are marked with braces.

Phase 1
iteration 1 [Completed - Apr 28, 2016]

- design the mechanical structure of the robot; assemble prototypes;
- setup local web server to receive IMU data sent from each robot;
- send moving instructions through Wi-Fi to each robot based on a naive, deterministic algorithm;
- assemble a four-robot system; the system is able to move steadily;

(new) iteration 2 [Completed - May 12, 2016]

- use new stable servo
- standardise model:
- easy to assemble
- legs are of the same length
- calibrate servo
- modify robot design to accommodate requirements of power supply; use flexible cable;
- check MCU code to solve occasional delays
- based on IMU to know directions
- work on the server program in preparation for phase 2;

Phase 2 [Ongoing]

- (new) build a long-lasting and safe playground;
- optimise the developed moving strategy based on ML algorithm;
- regulate the data received, establish the learning model;
- build a database to store relevant info.

- try more advanced ML algorithms to tune the robots;

Phase 3

- realize simulators on the host;
- without prior knowledge of how it is assembled, the system is able to develop satisfying optimal strategies through learning;
- support more interesting ways of assembling these robots;
- the system can move on different terrains;

We added an iteration 2 at the end of phase 1. It was because that what we have achieved in phase 1 was still inadequate for starting phase 2. When designing the milestone goals and deliverables, we deliberately set different focuses for each phase, but failed to see the fact that starting to work on a different focus needs sufficient preparation. Therefore, we added one extra iteration in phase 1, to refine the work done in phase 1 and prepare us for the goals of phase 2. We also added a goal in phase 2 to build a safe playground for the robots. It was motivated by an unfortunate lab accident before mid-quarter presentation, where our four-unit group robot fell from the table and hit the ground, maimed two precious robots at once.

There are several goals that we failed to meet.
First, at the start of iteration 2 we thought we may need a better servo. The original servo used had somewhat unstable and unpredictable performance from time to time. However, this problem was solved after we changed our power supply so that the servo gets enough voltage. This is an example of two seemingly unrelated problems that were actually generated by one cause. Furthermore, we planned to standardize the robot design to reduce the noise brought to the system. It soon revealed that manually calibrating all the physical components to the same length and shape is just too difficult, and this hardware inconsistency has to stay in the system and be overcame through learning.
Finally, we failed to implement reinforcement learning on our robot group. It turns out that only realizing very simple machine learning module requires us to build a polished platform and we simply did not have enough time to advance our algorithm.

5 Summary

During the ten weeks we have built Project Tau from scratch to a well-designed hardware and software platform, capable of generating intelligent moving strategies based on the sensor data received. At the end of this quarter, the hardware structure of the robot is reliable and strong enough to cope with different situations without deformation and vibration, creating a desirable learning environment for the group robot. The software platform is well-structuralized so that any future extension and advancement in algorithms can be applied without major changes in the framework. The group robot is able to infer its movement information based on the sensor data collected, and then decide its next step based on its experience.

We will continue to work on this project in the summer, in the hope of advancing what we have achieved in phase 2 and starting to work on phase 3. One of our major goals is to implement reinforcement learning algorithms on our robot. Furthermore, we will begin to simulate the group robot in order to collect massive sample for learning. We will also try to study some cutting-edge learning methods like policy learning, sparse learning to bring Project Tau to a new level.

References

Rubenstein, M., Cornejo, A., & Nagpal, R. (2014). Programmable self-assembly in a thousand-robot swarm. *Science*, *345*(6198), 795–799.

Shoham, Y., Powers, R., & Grenager, T. (2007). If multi-agent learning is the answer, what is the question. *Artificial Intelligence*, *171*(7), 365–377.

Weiss, G. (1999). Multiagent systems: a modern approach to distributed artificial intelligence.