Robotic Swarming

Will Ferenc, Hannah Kastein, Lauren Lieu, Ryan Wilson

August 5, 2011

Abstract

This paper describes the research conducted during the Summer 2011 Research Experience for Undergraduates at the UCLA Applied Mathematics Laboratory Swarm Robotics Testbed. The robotics team set out to extend the capabilities of the third generation autonomous vehicles by extending the on-board algorithm processing and support of sensor devices. The research project focused on advancing multi-robot capabilities, generating path planning and swarming algorithms to implement on the testbed. The team also laid the groundwork for camera implementation, and redesigned the peer-to-peer network protocol to enable inter-vehicle communication. Through collaboration with the University of Cincinnati Cooperative Distributed Systems Lab, the team set up Transmission Control Protocol/Internet Protocol (TCP/IP) communication in Matlab for inter-testbed cooperation.

Contents

1 Introduction

The development of cooperative behavior between autonomous robots is useful because of its applications to multiple fields. This technology is utilized at industrial facilities to heighten efficiency, and unmanned autonomous vehicles are also useful for mapping and exploration objectives. Small micro-cars such as those used in the UCLA Applied Mathematics Laboratory are capable of dispersing to explore different regions of unknown territory and reporting information on the surrounding environment. These robots can be used to scour the rubble after a natural disaster and survey the terrain without putting human lives at risk. As the capabilities of autonomous robots expand, their efficiency and extensive capabilities make them an important tool used in modern society.

2 The Hardware

2.1 Camera

This year we began work on the camera unit. Initially the camera was put on the car by Anteros Labs without any hardware or software support, only power connections. The camera now has hardware support for all pins and interrupts as well as power. IIC serial communications have been implemented with the camera. The serial communication is over pins 4 and 5, SCL and SDA (see Figure 1). This will allow configuration of the camera. Hardware to read the data is also in place. All that remains for a functioning camera is to setup the interrupt controllers for VD, HD and DCLK. These are located on Figure 1 as pins 8, 9 and 10 respectively. They are the pins that give the synchronizing pulses for each frame of the image. VD is the vertical sync pin. HD is the horizontal sync pin and DCLK is the data clock. Timing diagrams for the behavior of these pins is provided in the camera spec sheet on page 15. The interrupts must be compatible with whatever peer-to-peer algorithm is in place. Please note that before service all of the cameras must be checked for continuity and for voltage as there are connections missing on some of the boards. Also, the camera prototype vehicle has had a jumper placed from IOVDD to PVDD and the trace to IOVDD removed from the board. It is suspected, but not confirmed that the voltage on this pin was too high for operation. The voltage is outside the recommended operating range, but not outside of the safety margin. Continuity needs to be checked from the FPGA all the way to the camera pins. Figures 2 and 3 provided for the continuity checking operation, they represent the pin list and netlist conversion table respectively. The minimum steps to

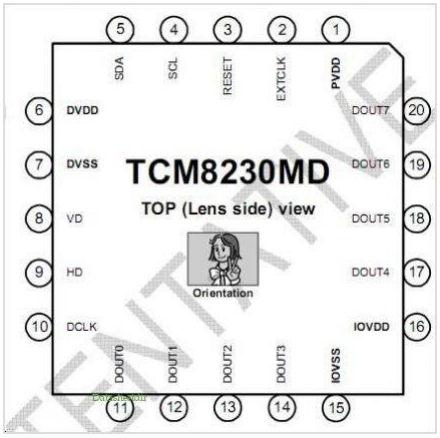

Figure 1: Camera pins

configure the camera are to use setupCameraGPIO, SetupInterruptSystem and InitCameraI2C. This runs the camera reset seuquence then the self test, interrupt setup serial initialization and finally serial configuration. Warning: there are still problems with the interrupt system, but the functions should return successfully.They are completely automatic and require no user input. These functions are described in detail in the in-code documentation.

2.2 Communication Systems

2.2.1 Upper Board Serial

The Upper Board Serial is the primary link for debugging, peer-to-peer and Internet connection. The UBS runs at 115200 baud with no parity bits. This is the serial link to the computers. This does not allow programming but does allow run-time communication. The robots connect to the wi232 devices using the UARTlite IP from Xilinx. Instances of MATLAB can connect to this network using the serial object from the instrument control toolbox with the appropriate wireless connection hardware. No other action is necessary because this is a simple point to multipoint network where there are no parity bits and no flow control. Through MATLAB you can write by simply using the fwrite command with the serial object and read with fread. Examples are provided in the form of the 2011 code.
Recommendation: Use either xil_printf or print, they are both large functions, so only use one if possible.
Check the MATLAB documentation for information about the serial object.

2.2.2 Lower Board Serial

No modifications were made to the lower board. Refer to the 2010 documentation for details about streaming from the Lower Board.

2.2.3 MATLAB Internet Connection Interface

The Ethernet system allows communication over the internet. The Ethernet communication is done through MATLAB. Two MATLAB instances (one at University of California, Los Angeles and one at University of Cincinnati) communicate data from testbed to testbed. Each testbed has its own method to transmit data from the MATLAB instance to the agent robots but the information over the network must be standardized. The easiest way is to order the information consistently. Examples of this can be found in the 2011 MATLAB code.
MATLAB uses the TCPIP object from the instrument control toolbox to set up a TCPIP server and client to transmit data from instance to instance. Fwrite and fread can be used just as in the case of the serial object.

3 Peer-to-Peer Networking

The 2010 robotics team set up a peer-to-peer network using a finite state machine in which an individual robot switched between four states: the tracking state, the transmission state, the receive state, and the calculation state. The transmission protocol for the network used a token topology. There were several problems with the peer-to-peer that prevented the network from being fully implemented and integrated into the testbed. Because of the transitions between the states in the finite state machine, if anything other than a complete message is received, the token is lost and the peer-to-peer stops functioning. Interrupt handles were implemented to prevent cars from getting stuck in the receiving state if a transmission error occurs. However, the cars only got stuck in an infinite calculation state instead of the receive state. As a result, the utility of this peer-to-peer network was limited because the micro-cars did not use inter-vehicle communication, but instead used the overhead camera tracking to determine the location of the other robots.
The communication network was structured in a way that caused message collisions, making the broadcast unintelligible to the cars. Inside the finite state machine, the cars switched between states independently, thus there was no guarantee that a car would be in the receive state when a message was about to be sent. Thus, sent messages had an increased probability of not getting received. Since missed messages meant lost tokens as well, one message error could disable the entire peer-to-peer network. The peer-to-peer communication structure needed to be redesigned in order to increase its robustness and flexibility.
Time-based scheduling was used to resolve the peer-to-peer network issues. The new configuration assigned specific time slots for broadcasting, processing, and

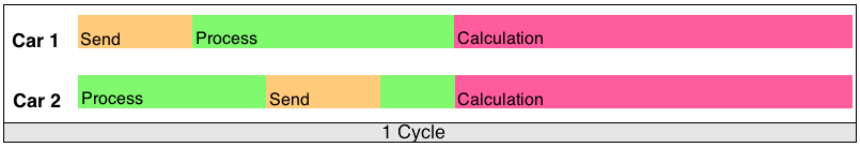

Figure 2: Example of peer-to-peer scheduling with two cars

calculation for each individual car. Without the token, the micro-cars would not get stuck in an infinite loop since time-based scheduling removed the systems dependence on receiving the previous message. By synchronizing the timers on every robot and scheduling broadcasting times to avoid radio broadcasting interference, they can rapidly share and update information without relying heavily on the lossy network.

3.1 Broadcasting Using Time-based Scheduling

The problems with the previous peer-to-peer network led to the restructuring of the system on a time-based schedule. Timer synchronization was necessary to ensure that the transmission cycles on each car begin at the same time. The decision was made to design the communication network to send smaller packets of information more rapidly instead of larger packets less often. This decision was based on the idea that even if one packet is corrupted or lost, another will be available soon enough that the loss would not negatively affect the system. Thus, during a broadcast, an individual sends a message consisting only of its own information. Each message broadcast contains the following: a designated header (one byte), the car identification number (one byte), the x-coordinate (two bytes), the y-coordinate (two bytes), the heading (two bytes), the IR sensor reading (two bytes), and a designated terminator (one byte). Thus, the total message size is eleven bytes. Looking at the upper board radio specifications, we know the transmission rate is 115200 bps. Using only 80% of that means a rate of 92 kbps. The time required to for one broadcast is thus 1.2ms. Allotting 1ms for processing for each message, means the maximum amount of time needed for broadcasting is 19.8ms (assuming there are nine cars on the testbed, the maximum number of cars available). The limiting factor for the cycle length is the rate at which information is transferred from the lower board to the upper board, 30 Hz. Allowing the cycle time to be 33ms thus leaves at least 10ms for algorithm calculations and commands a cycle. The car information is also updated before each message is sent out, capturing all the data without redundancy.
Thus with a cycle time of 33ms, the schedule is as follows:

- An initialization period to allow for schedule setup and broadcast time assignments. The initialization also includes timer synchronization across all of the cars. Broadcast times are scheduled to occur every 3ms.

- When not broadcasting, the cars wait in the processing loop for the message to be received. As a message comes into the radio, the interrupt handler is activated and the message is moved from the radios hardware buffer to the global array wi_Buff_UB. When the message is stored into the global array, the processing loop parses the message and stores it. The car then returns to checking its broadcast time against the timer and waiting for a message to be received.

- After each car on the testbed has broadcast its message, all the cars enter a calculation state. Thus, the fewer the number of cars on the testbed, the more time is allotted for calculations. It is during the calculation period that various algorithms can be inserted and executed.

- When the calculation period ends, all the cars reset their timers and repeat the above schedule.

3.2 Peer-to-Peer Communication Test: Following

In order to determine if the restructured peer-to-peer network was a more robust system that the previously implemented protocol, it underwent the following assessment on the testbed:

- Two cars were placed on the testbed. One was designated the leader, and the other was designated the follower.

- The leader was programmed to drive in a circle in its calculation period. The algorithm for the follower was to take the coordinates received from the leader and drive to that location.

The test was successful: both cars on the testbed behaved as desired. However, the functionality of the peer-to-peer network has not been thoroughly tested for a larger number of micro-cars due to time constraints. A preliminary swarm test was done with three cars, however it was unsuccessful.
Because the USB serial transmitter and receiver is on the same channel as the upper board radio on the car there is an issue with the serial picking up and displaying all of the unparsed messages that the cars send during peer-to-peer. Since there are eleven bytes per car being transmitted 30 times a second, the serial is essentially flooded. This makes it very difficult to use print statements to aid in the debugging process. The entire peer-to-peer cycle will need to be slowed down considerably in order to debug the problem that occurs when using more than two cars on the testbed.

4 Algorithms

4.1 Path Planning

The goal of a path planning algorithm is to follow a path to a predetermined target without running into other robots and barriers. A sufficient model for

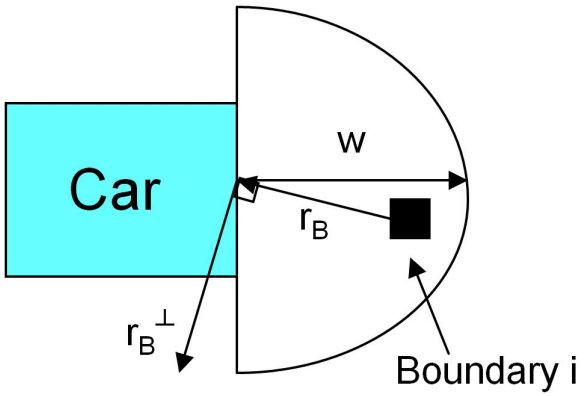

Figure 3: The robot detecting a barrier and feeling a force perpendicular to the barrier

this is a potential field, where each "object" emits a force. Other robots and barriers emit a repulsive force while the target emits an attractive force. The robot moves in the direction of the total force with constant velocity. The following equation governs this type of movement.

$$F = Q_r \left(C \frac{r_t}{||r_t||^2} + Q_r \sum_{i=1}^{K-1} \frac{r_i}{||r_i||^2} + \sum_{i=1}^{L} B_i \right) \tag{1}$$

Q_r is the robot potential, C is the target potential, r_t is a vector from the target to the robot, r_i is a vector from the moving robot to the i-th robot and B_i is the i-th barrier term. Each robot, barrier and target is assigned a constant potential, which corresponds to coefficients Q_r, B_r (which will be discussed soon) and C. Also, K is the number of robots on the testbed and L is the number of boundaries in the environment. The first two terms are simply a direct attraction or repulsion inversely proportional to distance. However, the boundary term is a bit different because we want the robot to go around the boundary instead of directly away from it. As shown in Figure 4.1, the robot feels a force perpendicular to the barrier instead of directly repulsing it. The robot only detects boundaries in the semicircle where the robot is facing, so this ensures the robot moves around the boundary.

$$B_i = B_r \left(\frac{r_B^{\perp}}{||r_B||^2} \right) \tag{2}$$

Similar to the terms in (1) , B_r is the boundary repulsion term and r_B is the vector perpendicular to the vector from the i-th boundary to the robot. However, deciding which of the two perpendicular vectors to choose is an interesting question. Choosing one direction for every robot means all robots go around

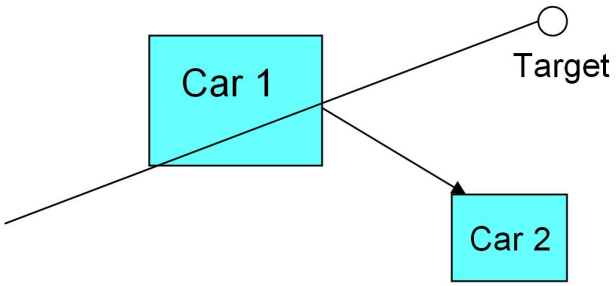

Figure 4: How the robots detect which direction r_B should point

the same side of the boundary, possibly causing a traffic jam or just not being very aesthetically pleasing. Ideally, we want half the cars to go around one side of the boundary while the other half go the other way. Drawing a line from the target to the robot, the testbed is split into two regions. One of the vectors perpendicular to the boundary lies in each region. Whichever region contains less other robots, the boundary vector lying in that area is used in the boundary term.

4.2 Leader Following with Swarming

The goal of this algorithm is for a group of robots to follow a leader robot without running into one another while maintaining formation. Again, a potential model is used, but this time the potential is exponential instead of linear. This is similar to Morse potential as seen in molecular physics. The following differential equations govern the velocity v_i and position x_i of each robot, except the leader which operates independently of the swarm.

$$\frac{dx_i}{dt} = v_i \tag{3}$$

$$\frac{dv_i}{dt} = (\alpha - \beta||v_i||^2)v_i - \nabla U(x_i) + \Sigma_{j=1}^{N} C_0(v_j - v_i) \tag{4}$$

$$U(x_i) = \frac{1}{2}C_l(x_i - y)^2 + \Sigma_{j=1}^{N} C_r e^{||x_i - x_j||/l_r} - C_a e^{||x_i - x_j||/l_a} \tag{5}$$

U is the potential function, N is number of robots on the testbed and y is the position of the leader robot with constants m, C_0, C_l, C_r, C_a, l_r and l_a. The constant m refers to the robot mass, C_0 is the velocity alignment coefficient, C_l is the leader potential coefficient, C_a and C_r are the robot attraction and repulsion coefficients, respectively, and l_a and l_r are the robot attraction and repulsion lengths, respectively.

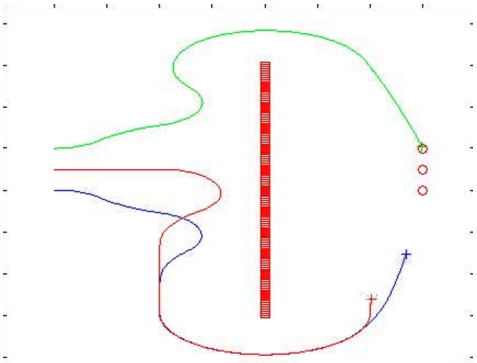

Figure 5: A simulation of the path planning algorithm with three robots

4.3 Simulations

Simulations for both algorithms were written in Matlab. For the path planning algorithm, the robots could detect boundaries in a semicircle in front of them with fixed radius w (as seen in Figure 4.1) to simulate a camera or infrared sensor. A boundary is represented by a point, so to create large boundaries, several boundary points are clustered together. Each robot had its own target, but the targets were placed near each other.
For the swarming algorithm, an ordinary differential equation solver was used to calculate the velocity and position of each robot at every timestep. The leader robot had a circular path and moves at a constant velocity.
For both algorithms, parameters were chosen by experimentation. The values that were used for the path planning algorithm are $Q_r = -150$, $B_r = -2000$ and $C = 1000$. In the swarming algorithm, the constants were chosen as such: $m = 1$ $C_0 = 1$, $C_l = 0.7$, $C_r = 50$, $C_a = 90$, $l_r = 12$ and $l_a = 2$. Using the Instrument Control Toolbox, communication between the University of Cincinnati Mathematics REU program and our team was established via TCP/IP. The swarming algorithm previously mentioned was implemented on two separate sessions of Matlab, one at each university. Position and velocity data from each local swarm was exchanged real-time, so the swarms could follow the leader robot together.

4.4 Implementation

The path planning algorithm was implemented on the actual robots. However, to detect boundaries, the robot would gather IR sensor data. Then, a curve was fit to convert the reading to an approximate distance away of an object. If the object is less than a certain distance away and is inside the testbed, then the barrier term is nonzero. The algorithm works on one robot, but it does not always take the most efficient path around a barrier. This is due to the fact

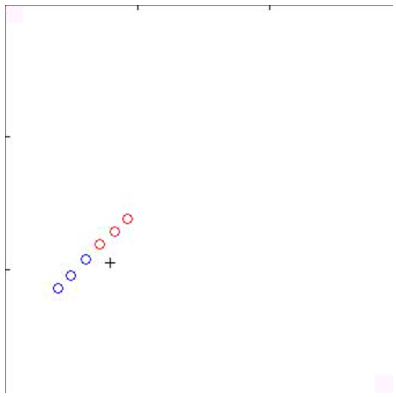

Figure 6: A simulation of the swarming algorithm with one leader robot, three blue robots from UCLA and three red robots from Cincinnati

that the IR sensor sees objects outside the testbed that are white or reflective, so naturally the IR sensor reading is far too large, especially near the boundary of the testbed. Hopefully, in the future, the camera will be used for on-board image processing to detect barriers more accurately.
Unfortunately, there are issues with the peer-to-peer communication, so multiple car path planning and swarming were not able to be implemented on the cars. However, the algorithms have been coded in C, so once car communication works, the algorithms should work as well. Naturally, this will allow for swarm-to-swarm communication with the Cincinnati team, using a Matlab-to-C parsing script which we already have written as well.

5 Conclusion

The robotics team developed effective communication systems that allow for cooperative algorithms between both testbeds and individual robots. The University of Cincinnati robotics team and UCLA Applied Mathematics Lab have coordinated swarming simulations, preparing for actual inter-testbed swarming maneuvers such as barrier avoidance and leader following. In addition, the camera hardware on the third generation micro-cars has been configured for the next step of writing the drivers for the device. The completion of these tasks lends the robots new sensing capabilities and establishing interaction between different robotics platforms paves the way for multi-robot coordination.

6 Recommendations for Future Work

The team has generated a list of the following recommendations regarding the equipment and operation of the UCLA Applied Mathematics Laboratory

testbed to facilitate the swarm robotics research conducted annually.

- Design the next generation of robots to have a sharper turning radius so that the vehicles can more accurately track a prescribed path instead of being limited to a turning radius of 25 cm, which makes up a sixth of the testbed length, or be capable of holonomic drive (mechanum or omni-wheels).
- Include an additional radio for the next generation of robots so that there is no interference between inter-vehicle communication and the information exchanged via the GUI interface. As there is now a high amount of traffic over the Upper Board radio for peer-to-peer communication.
- Complete and stabilize the image capture software. The interrupt and data collection routines need to be completed and integrated with the existing suite of peer-to-peer interrupts.

7 Acknowledgements

The team would like to thank Rick Huang for providing his assistance with the operations of the robotics testbed and troubleshooting the algorithms developed for the micro-cars during the research program. The team would also like to thank Jérôme Gilles for guiding them throughout the program, Yasser Taima for his help in working to implement his own swarming algorithms, and Andrea Bertozzi for creating the REU program and providing undergraduates with the opportunity to perform this research.
Also, the team thanks the UCLA Mathematics Computer Consulting Office for obtaining the needed Matlab Instrument Control toolbox to allow for TCP/IP communication and helping establish a method to work with the UCLA firewall security system. Additionally, the team thanks the University of Cincinnati Cooperative Distributed Systems Lab for their collaboration to develop inter-testbed communication.

References

[1] D.J. Bruemmer, Dudenhoeffer D.D., and J.L. Marbe. Dynamic-autonomy for urban search and rescue. *AAAI Technical Report*, 2002.

[2] K.C Cao, G. Xiang, and H. Yang. Formation control of multiple nonholonomic mobile robots. *International Conference on Information Science and Technology*, pages 1038–1042, 2011.

[3] B.S. Chlebus, L. Gasieniec, A. stlin, and J.M. Robson. Deterministic radio broadcasting. *Automata, Languages and Programming: Lecture Notes in Computer Science*, pages 1038–1042, 2000.

[4] J. Fredslund and M.J. Mataric. A general algorithm for robot formations using local sensing and minimal communication. *IEEE Transactions on Robotics and Automation*, pages 837–846, 2002.

[5] M. Gonzalez, X. Huang, D.S. Martinez, C.H. Hsieh, and Y.R. Huang. A third generation micro-vehicle testbed for cooperative control and sensing strategies. *International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2011.

[6] G.A. Jacoby and D.J. Chang. Towards command and control networking of cooperative autonomous robotics for military applications (carma). *CCECE/CCGEI*, pages 815–820, 2008.

[7] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g 2 o: A general framework for graph optimization. *ICRA*, pages 1–7, 2011.

[8] Y. Landa, D. Galkowski, Y.R. Huang, A. Joshi, C. Lee, K.K. Leung, G. Malla, J. Treanor, V. Voroninski, A.L. Bertozzi, and Y.R. Tsai. Robotic path planning and visibility with limited sensor data. *American Control Conference*, pages 5425–5430, 2007.

[9] K.K. Leung, C.H. Hsieh, Y.R. Huang, A. Joshi, V. Voroninski, and A.L. Bertozzi. A second generation micro-vehicle testbed for cooperative control and sensing strategies. *Proceedings of the American Control Conference*, pages 1900–1907, 2007.

[10] W. Liu, Y.E. Taima, M.B. Short, and A.L. Bertozzi. Multi-scale collaborative searching and swarming. *Proceedings of the 7th International Conference on Informatics in Control, Automation, and Robotics (ICINCO)*, 2010.

[11] J. McLurkin, J. Smith, J. Frankel, D. Sotkowitz, D. Blau, and B. Schmidt. Speaking swarmish: Human-robot interface design for large swarms of autonomous mobile robots. *Proceedings of the AAAI Spring Symposium*, 2006.

[12] C. Moeslinger, T. Schmickl, and K. Crailsheim. Emergent flocking with low-end swarm robots. *Proceedings of the 7th International Conference on Swarm Intelligence: ANTS10*, page 424431, 2010.

[13] B.Q. Nguyen, Y.L. Chuang, D. Tung, C. Hsieh, Z. Jin, L. Shi, D. Marthaler, A. Bertozzi, and R.M. Murray. Virtual attractive-repulsive potentials for cooperative control of second order dynamic vehicles on the caltech mvwt. *American Control Conference*, pages 1084–1089, 2005.

[14] A.F. Scott and C. Yu. Cooperative multi-agent mapping and exploration in webots. *4th International Conference on Autonomous Robots and Agents*, pages 56–61, 2009.

[15] H. Seki, Shibayama S., Y. Kamiya, and M. Hikizu. Practical obstacle avoidance using potential field for a nonholonomic mobile robot with rectangular body. *IEEE International Conference on Emerging Technologies and Factory Automation*, 2008.

[16] M. Sugisaka and D. Hazry. User interface in web based communication for internet robot control. *ICCAS 2005*, 2005.

[17] A. Turan, S. Bogosyan, and M. Gokasan. Development of a client-server communication method for matlab/ simulink based remote robotics experiments. *2006 IEEE International Symposium on Industrial Electronics*.