CONCEPT GENERATION AND SELECTION

Robot Racers Butterfly Sharpshooters Team

Colby Ballew
Melyssa Fowler
Dallin Hague
Alex Harding
Billy Jacobs
Jacob Young

Winter 2012

TABLE OF CONTENTS

Introduction

Overview

For the Robot Racers Senior Project we are to take an RC truck and program it to play the game of capture the flag (or some other game if decided). The truck is equipped with a camera in order to navigate its way around the course and see other trucks. It also has a gyroscope for detecting turning speed and a wheel encoder to detect velocity. These will be used for the control system. The truck communicates with a base station using wireless, where we will have a debugging GUI.

There are many design choices to be made on this project, however most of them are relevant only to one portion of the project over which one of us is the lead. Since these design decisions only affect one portion of the project independent of the others we decided to leave those decisions to whoever is in charge of that portion. However two design decisions are critical to more than one subsystem and are therefore outlined in this document.

Purpose

The purpose of this document is to lay out all of the decisions that went into our final choice for design on two critical designs. The first of these critical decisions is where to do the vision processing, since it will be the most computationally intensive part of this project. This will create a huge bottleneck if not done correctly and will affect the performance of all other subsystems. It is also important for the embedded system design because the question involves whether or not the images will be processed in hardware or software.The second major design decision outlined in this document will be how we plan on sending video back to the GUI for debugging purposes. This is important to communication because if not taken care of the video could easily hog the wireless, it is also important to the GUI designer. Most importantly it effects vision and navigation because it will be the main way each of those systems will be debugged. It also is critical to the entire embedded system design. Both of these designs are critical because they can affect all other systems, and thus must not be taken lightly. This document serves to outline why we made the decisions we did. Other less critical decisions are not addressed in this document.

Procedure

For each of the critical decisions we have outlined several different concept options we have come up with for the design. Then we decided on what time of metrics were important in our decision. We assigned weight to each of the metrics since some factors are more important and deserve more consideration than others. We then created a Design Cost Function

Matrix for each decision and scored each of the concepts according to each metric. The weighted sums of each concept are shown, and our decisions for each score are explained. The concept with the best score was the design we chose to implement. By creating Design Cost Function Matrices we have made impartial decisions for the critical designs based on the most important factors.

Body of Facts

Specifications and Facts

Create an autonomous robot that:

- Is able to interpret visual stimulus and use it to navigate a course
- Is able to make game strategy decisions according to its current state and other cars
- Is able to communicate to the game referee and other robot racers
- Is able to control steering and speed with a feedback loop

In order to accomplish this we will need:

GUI

- Be able to send ESTOP and start commands
- This will be used to retrieve real time debugging information
- Low Priority: video feed

Controls

- Proportional Integrative Derivative controller

Vision

- Integrate camera with our design to receive vision
- Interpret flags from other robot racers
- Interpret base station
- Interpret course

Navigation

- Create internal map of obstacles and base station
- Capable of following other racers
- Low Priority: Multiplayer- communicating between the team to build the map
- Low Priority: Multiplayer- communicate to team to maneuver and avoid getting hit

Embedded Systems

- Use EDK to build system
- Virtex 4, Dual PowerPC Processors
- Use processor bus systems- communicate between modules and bus
- Bridge two processors together
- Able to route signals separate from the bus

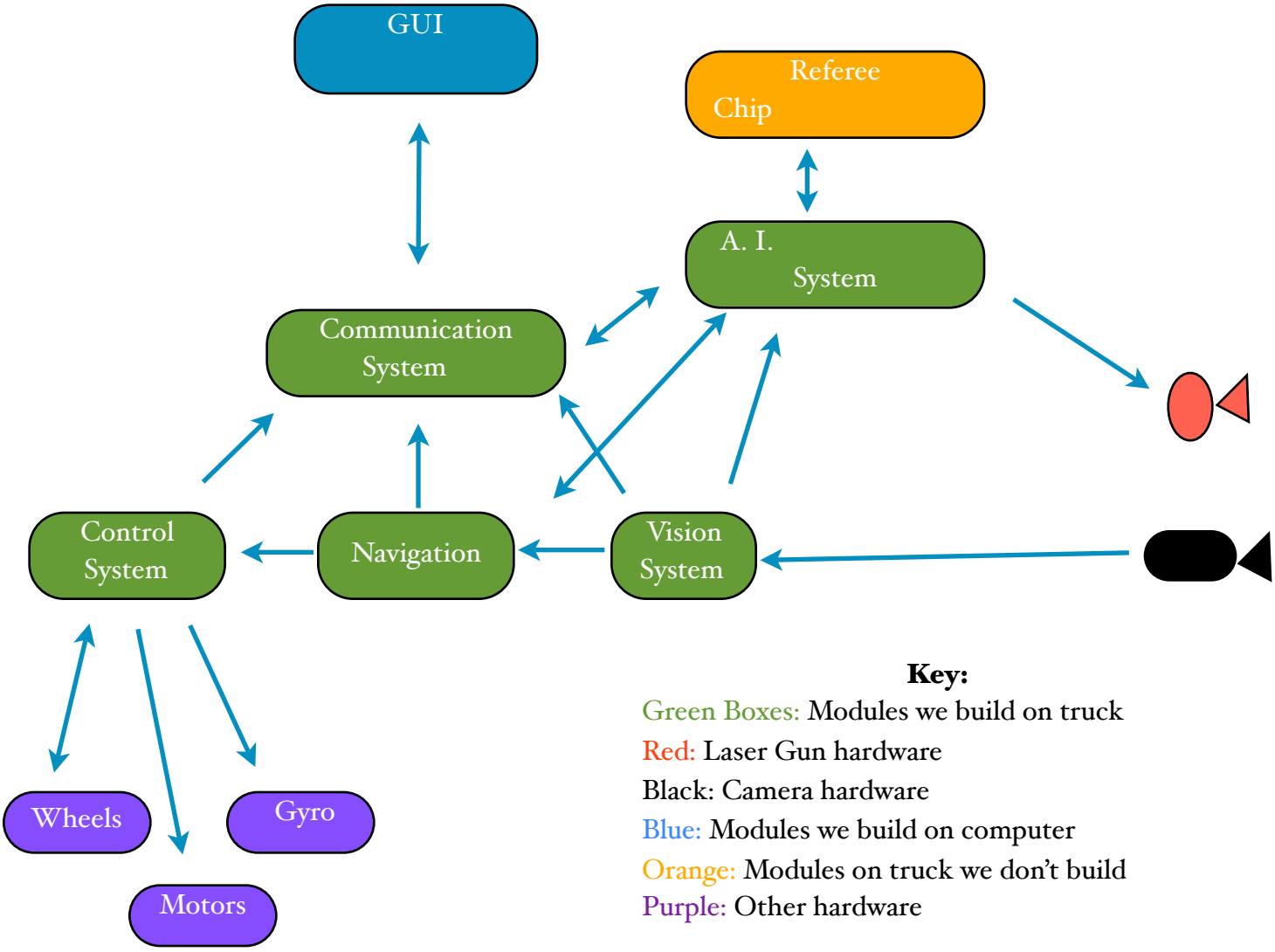
Communications
- Wireless communication between the car and the Computer/GUI
- Able to drive via keyboard

Game Algorithms
- Win
- Decide different priorities

Critical Assumptions
- Professors will decide final objectives within a reasonable time limit
- Hardware works to specifications
- Everyone follows our rules including quantity and quality of work
- Changes between single player and multi player in our design is reasonable (Hardware is the same, only software would change)
- We can use both processors effectively and dedicate one to vision
- When sending images across the wireless we can find a balance between the time it takes to compress one image and the time it takes to send another image.
- Certain compression algorithms are available for use

Proposed Design
As seen in the block diagram below, this is how the entire system is laid out. Images are received through the on board camera. This images are processed by the vision processor module which will interpret what it is seeing (e.g. enemy truck straight ahead) and send that data to both the navigation system and the game A.I. The game A.I. will decide what to do with that information and send commands to the navigation system. The navigation system will also use the vision data as part of a feedback loop to correct its course to where the A.I. has last directed it to go. The navigation will also send information back to the A.I. to indicate if it is in range of base stations or enemies. The navigation then uses its feedback loop to give velocity and turning radius commands to the control system. The control system uses a Proportional Integrative Derivative controller to send commands to the PWM of the motors to control speed and sets the steering angle of the wheels. At also reads from the wheel encoders and gyroscope in order to fine tune its velocity and turning radius. Another important module is the wireless communication module. All modules send data to this module if

needed for debug. This wireless module communicates with the GUI on the computer which can display text for debugging as well as a live video stream. It also receives the GO and ESTOP commands from the GUI which tell the game A.I. when to stop and when to abort. The game A.I. additionally controls the laser gun and communicates with the referee system. Each module is designed and managed by a different member of our team and it is up to them to make design decisions that affect their module only. The decisions that affect all modules are made as a group and are in this document.



Key:

Green Boxes: Modules we build on truck
Red: Laser Gun hardware
Black: Camera hardware
Blue: Modules we build on computer
Orange: Modules on truck we don't build
Purple: Other hardware

Concept Selection and Scoring

Vision Processing

The most computationally intensive portion of the product will be taking the images from the camera, adding filters to them and having the computer translate the images into objects such as other trucks, base stations and obstacles. This creates a bottleneck in the processing time, but is still critical for the Robot Racer to navigate the course and accomplish gameplay objectives. Without an efficient way to handle the vision processing, our Robot Racer will not have a competitive advantage.

Concept Definitions

Separate Processor - Have a single PowerPC processor handle only vision. The Virtex 4 board we will be using has two PowerPC processors. We would dedicate an entire one to image processing and have the other one use the RTOS to control all other truck functions.

Separate Processor + Hardware Preprocessor - The same idea as having a separate processor for vision, but also create custom hardware modules to apply filters before the images reach the processor. This would remove some of the load off of the processor.

Custom Hardware Module - Build a hardware module that can process the images without the need of software. This removes the entire load of vision processing off of the PowerPC.

RTOS - Have the vision processing handled in the RTOS along with all other truck functionality.

RTOS + Hardware Preprocessor - The same idea as having the RTOS do the vision processing, except it would also have custom hardware modules to apply filters to take some of the load off of the processor.

Design Evaluation Metrics and Weighting

Time to Implement - This includes the number of man hours we expect the design to be implemented in, and factors in the complexity of each solution. Because we have only until April to complete this project we weighted this metric at 30%, since it is very important and critical to our success on the project.

Speed - This is the time it takes to processes a single frame. This is important because you need to receive enough images quick enough to make real-time decisions for navigation. However we did not feel that it was as important as the time to implement so we weighted it at 20%.

FPGA Fabric Space - This is the amount of space on the FPGA that the vision module will take up. If the design is too large it will inhibit the amount and size of other modules we wish to create. Since at this time we do not anticipate the need for more custom hardware besides vision this is weighted at 10%.

Quality of Results - This is the quality of the processed images and vision interpretation and its usefulness to navigation and game strategy. Because we ultimately want to win the competition by having the best vision and navigation systems we weighted this at 40%

Design Cost Function Matrix

The options are each ranked between one and five, with one being the worst and 5 being the best. Our decisions are shown in the table below and explained in the next section.

Weight	*Metrics*	*Separate Processor*	*Processor + Hardware*	*Custom Hardware*	*RTOS*	*RTOS + Hardware*
30%	Time	3	2	1	3	2
20%	Speed	2	3	5	1	2
10%	FPGA Space	5	3	1	5	3
40%	Quality of Results	5	4	3	5	4
100%	Weighted Total	3.8	3.1	2.6	3.6	2.9

Results

Separate Processor - The time to implement this would be moderate since it includes a lot of software, for this reason we assigned it a 3. The speed of this solution would be a little bit slow because it would be dependent on the clock frequency of the processor and the efficiency of the software so we gave it a 2. The FPGA space it would occupy is none, since it is an all software solution so we gave it the highest score of a 5. The quality of the results would be very high as well because we would be able to fine tune the algorithms and implement many filters.

Separate Processor + Hardware Preprocessor - Because this solution would take a little bit longer to implement than just a separate processor design (because we would have to build the hardware module as well) we gave it a slightly lower score of 2 for time to implement. The speed of this solution would increase though because custom hardware in inherently faster than software and we would be able to dedicate less processor clock cycles to the vision processing. For this we gave it a 3. The FPGA space would increase but not as much as a full hardware module so we gave it a 3. The quality of the results would however deteriorate as there are greater limitations in hardware, so we gave quality a 4.

Custom Hardware Module - The custom hardware module received a 1 in time to implement because designing custom hardware is a lot harder than writing software and would potentially require more time to debug as well. However an all hardware solution would be much faster than software so it received a 5 in speed. It would take up the most FPGA space so we gave it the lowest score of 1. For quality of results it received a 3 because the hardware would have to be very basic in order to get it working within our time constraints.

RTOS - The design of using the RTOS to also do the vision received a 3 for time to implement. The time it would take to write the software would be the same as if we were writing it for its own processor. For speed, the RTOS solution received the lowest score of 1 because the vision processing tasks would have to compete with all other tasks the truck would need to do, i.e. Navigation, communication, A.I., etc. The design would take up no FPGA space and thus receives a 5 in this area. It also would have the same quality as the separate processor option because it is all software.

RTOS + Hardware Preprocessor - The time to implement for this would be slightly greater than the just RTOS solution because we would also have to design and build a hardware module. It is thus given a 2. The speed would however take a slight increase because more of the task would be put on a dedicated hardware module instead of being shared among the other RTOS tasks. It would take up the same amount of FPGA space as the separate processor with hardware preprocessor and thus receives a 3. The quality of results would similarly be degraded down to a four.

After adding up the weighted totals we found that the separate processor solution with no hardware preprocessing would be the most logical solution. This will be the design we will implement, and if we decide later that it is slower than we want and we still have time we can add the hardware preprocessing.

Video for GUI Interface

As seen previously, video is expensive to process. Another issue we needed to find a solution for was what was the best way to transmit the video to the GUI over the wireless without

causing a bottleneck and getting the best results. The video sent to the computer will be used to debug and see what the truck is seeing.

Concept Definitions

Send the frames "as is" - This solution is just to dump the image frames that are received from the camera straight across the wireless. This is our base for comparing all other solutions against.

Color Compression - This design is to compress the images before sending them by compressing the color data down to a smaller size.

Smaller Images - This design is to compress the image's size down and send smaller images instead of full size images.

Preprocessed Images - This design is to send images that have already been preprocessed by the vision interpretation software.

Preprocessed Smaller Images - This design is the same as the previous one, except that we would scale the preprocessed image down as well.

Design Evaluation Metrics and Weighting

Frames Per Second (FPS) - This indicates the number of frames per second we would be able to send over the wireless without it overloading. This is important because we want real-time images of what the truck is seeing and if there are too few frames per second the video we receive will be jumpy. Since we feel this is important for debugging we assigned this a weight of 30%

Time to Implement - This includes the number of man hours we expect the design to be implemented in, and factors in the complexity of each solution. Because we have only until April to complete this project we weighted this metric at 30%, since it is very important and critical to our success on the project.

Quality of Images - This is the quality of the images received after being sent. The quality isn't as important as long as they are still able to be interpreted and used for debugging, so we gave this a weighting of 10%

Usefulness of Images - This tells us how useful the images are for debugging purposes. It is not the same as quality because some preprocessed images may be useful but of low quality. Since this is ultimately what we need to debug, we gave this a weight of 30%.

Design Cost Function Matrix

The options are all based off of the baseline of sending "as-is" images. Lower (negative) numbers mean that it is worse and higher (positive) numbers mean that it is better. Our decisions are shown in the table below and will be explained in the next section.

Weight	*Metrics*	*As Is*	*Color Compressed*	*Smaller Images*	*Processed*	*Processed, Smaller*
30%	FPS	0	1	3	3	5
30%	Time	0	-1	-1	-1	-2
10%	Quality	0	-1	-2	-2	-3
30%	Usefulness	0	0	-1	5	5
100%	Weighted Total	0	-0.1	0.1	1.9	2.1

Results

As Is - Since we would not have to do any additional work besides dump the images across the wireless, this is our baseline so we set all values to zero in order to compare against it.

Color Compression - This would have slightly greater frames per second than the baseline since it would not have to be sending as much data, os for this reason we gave it a 1. Time to implement would be slightly worse than as is because we would have to implement a compression algorithm. However since there are many algorithms available for use this shouldn't be too much harder to implement so we gave it at -1. As far as quality goes there'd be a slight deterioration in quality so we gave it a -1. We still think it would be as useful as the normal image because having a large variety of color is not important to us since we'd be looking for blocks of all the same color. For this we gave it a zero.

Smaller Images - Cutting down the size of the images significantly reduces the amount of data we'd be sending and we'd be able to get a much higher FPS. For this we gave it a 3. The time to do would be comparable to the color compression, since we'd be throwing out every other pixel in each direction and this wouldn't be hard to implement so we also gave it a -1. For quality we would lose a little bit more than color compression because we would lose a lot of detail so we gave it a -2. The usefulness would also be decreased because a smaller image is hard to judge.

Preprocessed Images - Sending a preprocessed images would also increase our frames per second significantly because it would essentially be extremely color compressed since it segments the image into basic color blocks and edges. For this reason we gave it a 3. It wouldn't take that much longer to implement because the vision software is already preprocessing it, we would just have to convert that data back to an image. We gave this a -1. As far as quality goes it would be a -2 because we are losing all detail except what the algorithm detects as large color blocks and edges. However we gave it the highest rating for usefulness because it allows us to not only test navigation but see how the vision algorithm is operating and what the truck "thinks" it is seeing.

Preprocessed Smaller Images - For the preprocessed images we could get a much higher FPS since we'd be send very little data for each image. This is why we gave it a 5. The time to implement it is slightly worse than the last one since we'd have to implement both a size compression algorithm and convert the preprocessed data into an image. For this reason we gave it a -2. The quality of the image would be significantly worse and so it received a -3. The usefulness of this image would be the same as the preprocessed images since decreasing the size will not remove any detail that is useful to use, since there essentially is no detail in a preprocessed image. This is why we gave it a 5.

As seen in the matrix and reasoning above the best choice is to send preprocessed smaller images. This is what we have decided to implement.

Conclusion

We have chosen to implement the vision processing system by devoting an entire processor to image processing and using no hardware preprocessor. For transmitting video to the debug panel we have decided to send size compressed preprocessed images. The most significant factor in both of these choices was the time it takes to implement and the usefulness or quality of results. The reason these are important is because we are short on time and need to have the most effective solutions in the shortest amount of time. If we were given more time however we would probably try and optimize more and create more involved designs.

We believe that these two decisions were critical as they affected almost all of the team's individual areas (i.e. Navigation, GUI, Embedded System, etc.), and thus needed to be made collectively. Most of the other decisions are individual to each team member and can be made in a similar fashion on their own. However if other decisions that need to be made as a group come up, this document can serve as a model on how to do that. Furthermore, if changes are proposed for these designs, this document can show us why we made these decisions and be a good explanation to our customers.