Net-Scale Technologies, Inc.
281 State Hwy 79, Morganville, NJ 07751-1157, Tel. 732-970-1441, http://www.net-scale.com

DARPA-IPTO
Arlington, VA

Autonomous Off-Road Vehicle Control Using End-to-End Learning

FINAL TECHNICAL REPORT

Version: 1.2
July 30, 2004

This material is based upon work supported by the Defense Advanced Research Project Agency Information Processing Technology Office

ARPA Order No. Q458

Program Code No. 3D10

Issued by DARPA/CMO under Contract #MDA972-03-C-0111

Abstract

Autonomous ground vehicle control has vast commercial and military potential benefits but the performance of current systems is not sufficiently robust to allow deployment. Of particular difficulty is dealing with unknown environments, noisy sensor data and self-learning to optimize performance while driving. Advances in machine learning for pattern recognition, control problems, and, in particular, end-to-end learning promise improvements in these areas. The purpose of this project was to perform a short term and strongly focused effort to explore the benefits of end-to-end machine learning for this application. A test vehicle was built, consisting of a commercially available radio controlled model truck which was equipped with two light weight video cameras and other sensors. The vehicle's task was to drive on unknown open terrain while avoiding any obstacles such as rocks, trees, ditches, and ponds. The entire task was learned by the machine from examples. No hand designed algorithms have been used nor has the resulting network been hand tweaked in any way. The trained vehicle was able to successfully navigate through complex and tight paths between obstacles. A skill which in one case a human driver could not repeat even after multiple tries. The entire project was carried out in seven months by a team of three to seven people with a total hardware cost of about $30,000 including the vehicle, sensors, remote control and training and data storage computers.

Table of Contents

1. Introduction

It is not hard to explain the potential benefits of autonomous vehicle control for both, military and commercial applications. The benefits range from finding and assisting victims of an earthquake or on the battlefield, or operate in dangerous environments such as mine fields, to exploring foreign planets. It is therefore a widely studied field but practical applications are still very limited. A number of tasks are particularly difficult for traditional autonomous vehicle control, for example,

- Dealing with changing environments or operating in new and unknown environments.
- Learning while in operation or on a mission. This means being able to automatically vary a given approach to a new problem and when successful, learn from it so the knowledge can be put to use for similar situations in the future.
- Dealing with noisy and nonideal sensor data.
- Dealing with changing environments and environmental conditions such as lighting.
- Reconfiguring a system to operate for a different application or goal. This typically involves time consuming and expensive hand tweaking of the algorithms by human experts.

The idea of applying machine learning to the task promises to provide better solutions, in particular for the above problems. Machine learning is not a new field and other attempts have been made in the past to apply it to autonomous vehicle control (see for example [1], [2]). However, advances in machine learning for pattern recognition, control problems, and, in particular, end-to-end learning combined with the increased memory and processing power of today's computer systems promise a potential breakthrough. To understand the current state-of-the-art better and to help optimize future research grants in this area, DARPA-IPTO conducted this short term and strongly focused effort to evaluate the above promise.

Machine learning promises two particular benefits:

- Learning from example which can be used to retrain a vehicle to a changing environment without requiring any algorithm tweaking by human experts.
- Relearning while in operation to fine tune the performance and to potentially learn how to cope with new and previously unknown situations.

In this project we have chosen to use a commercially available radio controlled model truck as the vehicle which we equipped with two lightweight and low cost video cameras. Compared to the more expensive and high quality hardware typically used for such research this approach is very low coast. We did not have to worry about loosing the vehicle and could therefore make more aggressive driving experiments. Furthermore, the vehicle is relatively easy to reproduce and could in principle be used by many different small research teams. Furthermore, the whole setup is small enough to fit into a car and can be setup and operated by a single person. This provided for a very flexible environment for data collection and test runs. Finally, due to the low cost nature of the sensors and wireless data transmission hardware, the network was forced to deal with less than perfect sensor data.

The organization of the project was as follows. First the vehicle hardware was purchased and modified and then integrated with the network training environment

(Lush). Several weeks of data collection by a human driver followed after which the first training experiments were conducted. The initial training experiments with supervised learning were conducted off-line by the computer. The trained network was then installed on the computer which remote-controls the vehicle to conduct real outdoor test runs. In a later phase, the vehicle would collect new training data while it was driving itself with the previously trained network. A human driver was still present but would only override the network driving if the network made really bad decisions. The additional training data was used to retrain the network using reinforcement learning. Throughout the whole project the vehicle hardware and sensor data transmission was constantly being improved based on the field experience and data collection continued. The training data collected therefore continued to improve in quality. Of course, it would be preferred to use only training data of the highest quality but the time frame of the project did not allow to collect a sufficient amount of such data.

Supervised learning worked surprisingly well, especially when considering the poor quality sensor data. Reinforcement learning worked as well but did not show significant improvements over supervised learning. It is expected that sensor quality and small amount of available training data contributed to this result and one cannot necessarily conclude that reinforcement learning brings no advantage.

The input data presented to the network came directly from the sensors. No preprocessing other than subsampling the input images was performed. The network output was interpreted directly as steering wheel angle of the vehicle. No traditional processing algorithm was used nor were any weights of the network hand tweaked. The entire learning process was fully automatic with no human interaction and solely based on the available training data.

2. Training And Test Environment

The goals for the vehicle included not to use GPS because it does not work reliably in wooded areas and to drive entirely autonomously without any communication to a home base because such communication could be detected or disrupted by the enemy. Note, that this project depended heavily on wireless data exchange but only because the vehicle was too small to carry the computers and necessary power sources.

Figure 1 shows the training environment. It consists of an area of outdoor terrain in the size of about 50×50m, limited only by the range of the wireless data transmission. It is in open outdoor terrain and contains arbitrary natural and man made obstacles for the vehicle, such as trees, rocks, and walls. The goal is for the vehicle to run in an outdoor area which it has never previously seen, keep running in one direction but drive around any obstacles.

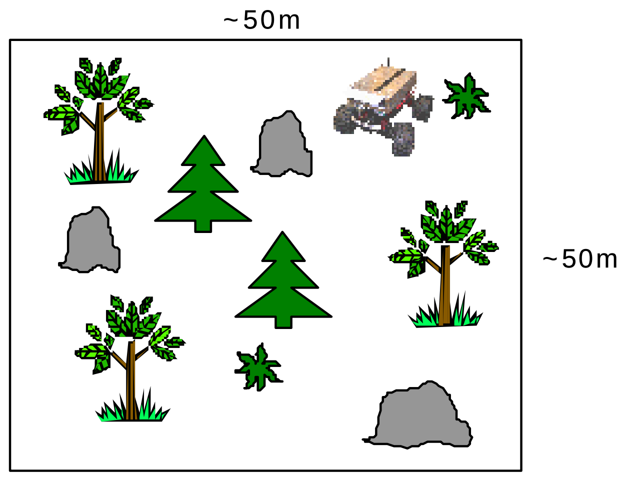


Figure 1. The "Playground" consists of an area of outdoor terrain in the size of about 50 ´ 50m. It is in open outdoor terrain and contains arbitrary natural and man made obstacles for the vehicle, such as trees, rocks, and walls.

3. Vehicle Hardware

3.1. Overview

The hardware for this project comprises several elements summarized in Figure 2. A remote-controlled vehicle collects sensory data (video, position and distance to objects) and transmits the collected data back to a PC workstation. The vehicle is driven using a joystick, whose actions are recorded by the PC program and sent to the vehicle via a custom module (see Section 3.5 for more details).

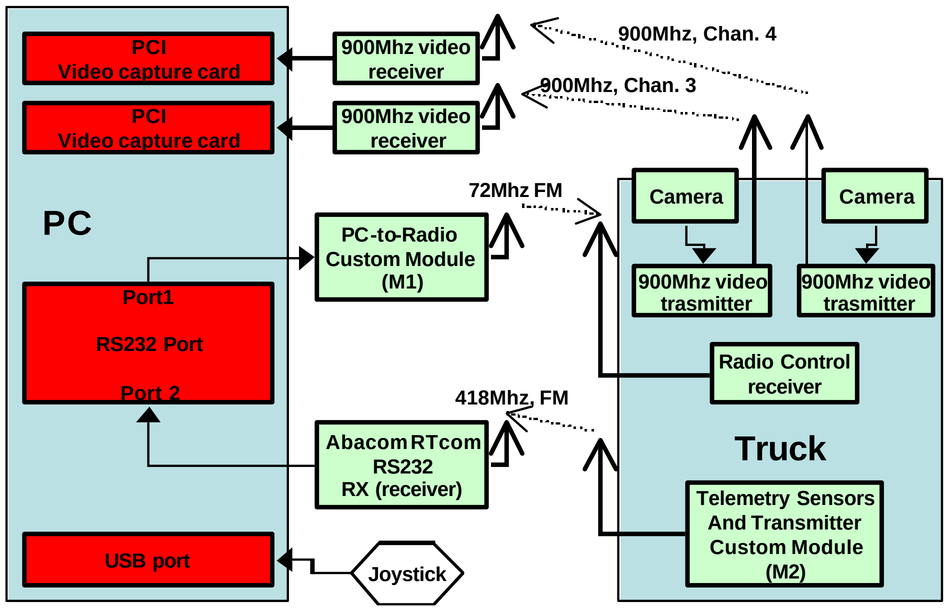


Figure 2. The overall architecture of the Autonomous Vehicle System. The submodule M1 is described in Section 3.5 and M2 in Section 3.4.

3.2. The Vehicle

The vehicle built for this project was nicknamed DAVE, for DARPA Autonomous Vehicle. The design for DAVE arose from the following constraints:

- Must be off-road capable.
- Must be able to carry a payload of at least 2 lbs.
- Must be built from off-the-shelves components to keep cost low.

Chassis

For the chassis, we used a commercially available 1/10th scale electric remote-controlled 'Monster Truck'. Figure 3 shows a view of the chassis.

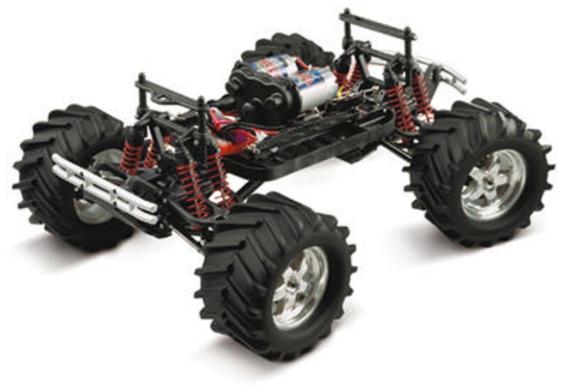


Figure 3. The chassis of the E-MAXX 1/10th scale 4WD Monster Truck from Traxxas (www.traxxas.com).

The Chassis is ideally designed to accommodate onboard electronics for sensory inputs. The chassis is 19 inches long (bumper-to-bumper), 16 inches wide (edge of tires) and 9 inches high (floor to posts, suspensions up).

Replacing The Remote Control And Receiver

To our knowledge, no off-the-shelve system exists that allows a PC program to control such a remote-controlled vehicle. One of the components of our custom interface system is an airplane remote that features a "trainer input" (these are used to train students to fly remote-controlled airplanes with a teacher being able to take control of the student's remote via a cable connecting the two remotes). Unfortunately the remote coming with the truck does not have such a capability and hence has to be replaced. The two remotes operate on different frequencies and hence the receiver on the vehicle has to be replaced as well. The receiver of the airplane remote is larger and does not fit inside the chassis in the same spot as the vehicle receiver. Figure 4 shows the two receivers and the assembly used to house the new receiver.

A piece of styrene is screwed on the chassis using the two holes visible on the right side of the original receiver. The new receiver is attached to the plate using Velcro.

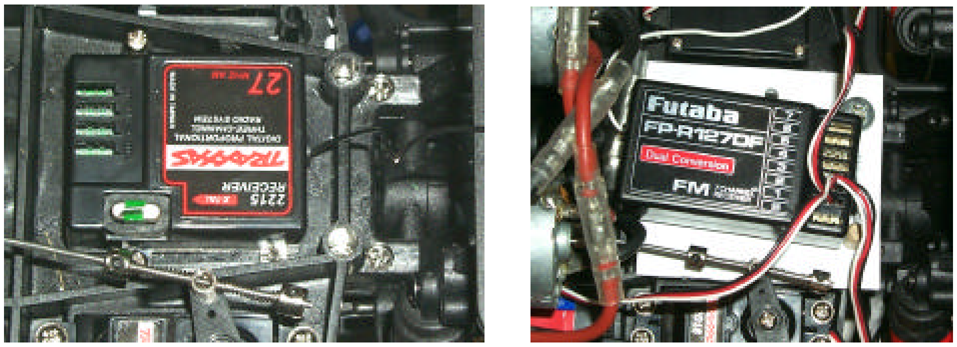


Figure 4. The original receiver (left) functions on the 27Mhz band and needs to be replaced by the receiver of the airplane remote (right).

Shielding the receiver

Experience has shown that the airplane receiver is prone to interference from both the motors and the wireless video transmitters. We have been using aluminum foil to provide adequate shielding. Aluminum foil is folded to make a 10-layer thick insulation that is wrapped around the receiver, letting only the servo and antenna cables stick out.

Changing The Gear Ratio to Slow Down the Vehicle

These 'Monster Truck' vehicles are designed for speed. They can go as fast as 30 mph. In our application, we rather need the vehicle to go at slow speeds (order 1-2 feet per second). To achieve this kind of speed the gear ratio needs to be changed from the default gear that comes with the vehicle. Figure 5 shows a view of the inside of the gearbox. Both the pinions and the spur gear can be purchased from the same manufacturer.

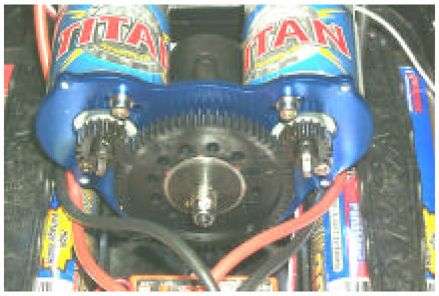


Figure 5. The E-Maxx gear box.

Factory installed is an 18 teeth pinion with a 66 teeth gear. We replaced them with a 12T pinion and 72T gear, providing a 66% decrease in speed (factor 1.6 slower). To change the pinions and gear, remove the gear cover and unscrew the pinions. The 1.5mm L-wrench provided by Traxxas might not be strong enough, buy a high quality reinforced wrench (e.g. by Bondhus). Next, remove the spur gear. Remember the position of the nut before unscrewing it. Transfer the clutch pegs to the new gear and reassemble. Try not to touch the pinion screws so as not to transfer oil onto them, which will make it more likely the come out during operation.

Box Housing For Onboard Electronics

A sturdy box was built out of 1/8th inch styrene plastic, reinforced with brass angles. The box is attached to the chassis using the four poles visible in the picture of Figure 3. Styrene is easy to cut and glue. The brass angles are cut to fit all angles of the box, while the plastic makes the walls of the box. The assembly is screwed together using six small (5-52) screws and nuts per face of the box. The front windshield is made out of clear Lexan to allow the camera to be positioned inside the box. Care must be taken that the camera lenses do not touch the Lexan, because this would result in the glass of the lenses eventually scratching the (less hard) Lexan and blurring the center of the image. The box (see Figure 6) has the following dimensions: length: 15'', width: 8'', height: 3''. The raw sheet of styrene measures 21 x 8 inches, hence the width of 8 inches. The box was dimensioned and positioned so as to give a 2-inche clearance between the bumpers and the edges of the box to avoid the box being directly hit in collisions.

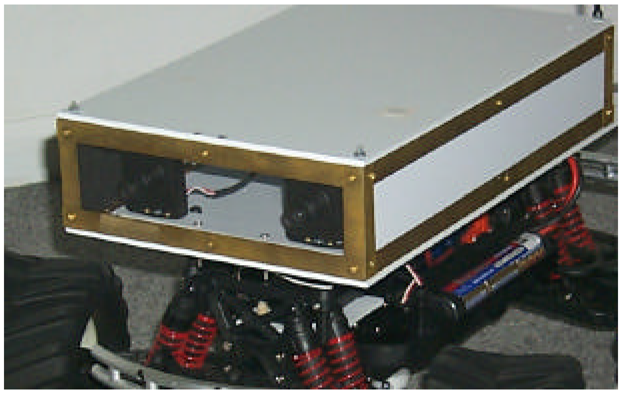


Figure 6. View of the Box housing the onboard sensory electronics.

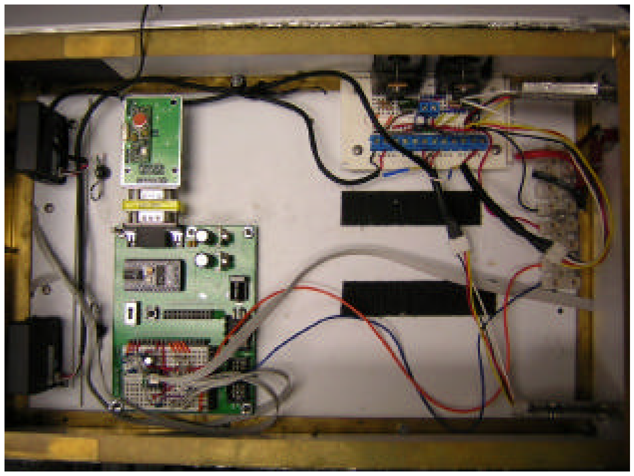


Figure 7. The inside of the vehicle's box housing the cameras, wireless video transmitters and sensors.

To minimize interference between motors, wireless transmitters and the vehicle receiver, the box is shielded using a layer of self-stick aluminum tape. Figure 7 shows a view of the inside of the box.

3.3. Stereo Cameras and Video Transmitters

Cameras

In order to provide good quality images, we stayed away from pinhole type lenses as well as CMOS imagers. Also, most of the cheap CMOS, pinhole-style cameras have no good automatic backlight compensation, resulting in the image dramatically darkening as a bright spot appears. Another important factor is the availability of wide-angle lenses. For navigation, the robot needs at least a 100-degree field of view. Most small CCD cameras have 1/3 inches CCD (measured diagonally), hence the camera lens must be a maximum of 2.5mm focal length.

Mount For Cameras

The two cameras need to be mounted on a separate assembly, so that the entire assembly can be removed from the box without having the cameras move relative to each other. This is necessary because the network must be able to learn to extract 3-D information from the stereo view provided by the two cameras. If the cameras would move relative to each other, the network would have to relearn their geometry and the previously collected data could not be used anymore. Figure 8 shows the assembly that holds the cameras and Figure 9, the two cameras on their mount inside the box. Specific dimensions of the assembly can be found in Appendix 11.6.

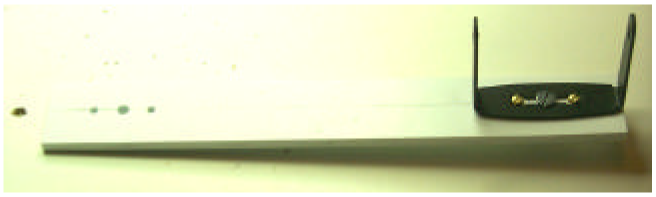


Figure 8. The mount for the stereo cameras.

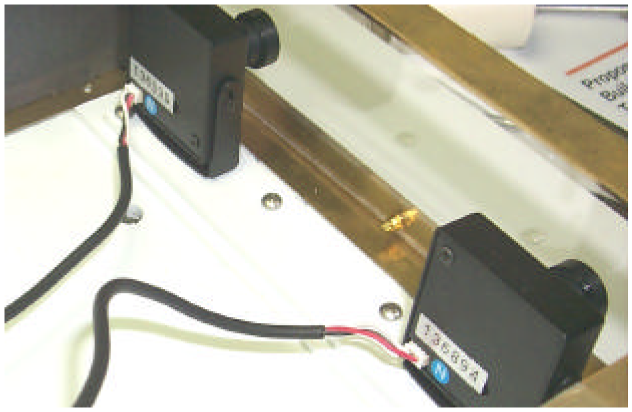


Figure 9. The two cameras mounted.

Wireless Video Transmitters

Off-the-shelve video transmitters are available at different frequencies. An initial choice of the 2.4-Ghz band proved unsuitable for the vehicle because of frequent dropouts in the video as the vehicle was moving. A second implementation using the 900-Mhz band resulted in more stable video, albeit still showing occasional dropouts. Quality was later improved using a larger antenna for the receiver. Another constraint in the choice of these video transmitters was the need for two separate channels to transmit the video from the two cameras. Most 900-Mhz video transmitters are locked into channel 1 of that spectrum, however some companies sell transmitters operating on 4 channels, this needs to be verified before ordering the parts. Video transmitters are available in a range of power from 100 mW to 1 W. We struck a compromise between power consumption and transmission range and chose 500 mW. The video transmitters are attached to the back of the vehicle's box.

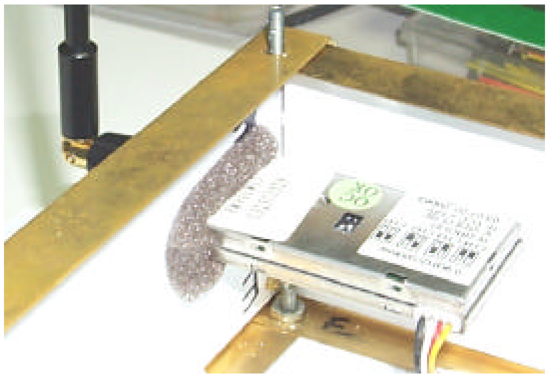


Figure 10. One of the wireless video transmitters.

Figure 10 shows how the antenna sticks out of the box, while the body of the transmitter is inside. A single hole drilled in the box allows the screwed-in antenna to secure the transmitter to the box. A schematic showing how the transmitters and cameras are connected is shown in Appendix 11.4.

Power Supply

The total power consumption of the onboard electronics is about 15 W at 14.4 V. Power for the onboard electronics comes from the battery packs of the vehicle. Each pack supplies 7.2 V (6 cells of 1.2 V each). There are 2 packs, thus providing 14.4 V. The cameras and video transmitters need a regulated 12 V supply. Hence the need for the assembly shown in Figure 11. It contains two voltage regulators and associated electronics. Two radiators are attached to the regulators to avoid over-heating. A detailed schematic of the power supply can be found in Appendix 11.4.

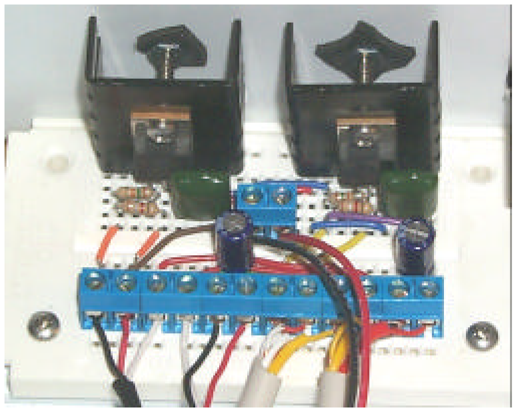


Figure 11. The Regulated power supply assembly for the onboard sensory electronics.

3.4. Compass, Tilt and Distance Sensors

A variety of sensors can be connected to the vehicle. A micro controller takes measurements from the installed sensors, makes necessary conversions and sends the data stream wirelessly to the PC. We chose to implement a compass, tilt and distance sensors to provide the network with feedback during the unsupervised learning phase. The schematics for the onboard sensors can be found in Appendix 11.4. The program listing for the micro controller can be found in Appendix 11.1. Figure 12 shows a view of the micro controller board with the sensor cables attached to the breadboard area. As a micro controller we use the BS2SX Basic Stamp from Parallax Inc. We use a convenient development board [7] featuring a breadboard area, a 5-V voltage regulator and a serial RS232 connector, also available from parallax.

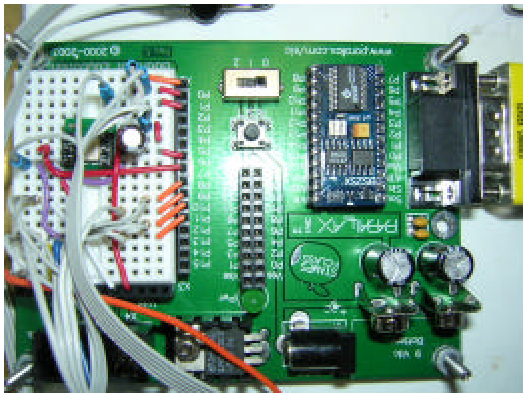


Figure 12. The micro controller boards used to read sensors.

To transmit the sensor readings, a wireless RS232 link is used. We use the RTcom-RS232 (Rx/Tx) from Abacom Technologies [6]. This board has a female DB9 connector and can be attached to the female DB9 connector of the micro controller board using a DB9 gender changer as shown on Figure 13.

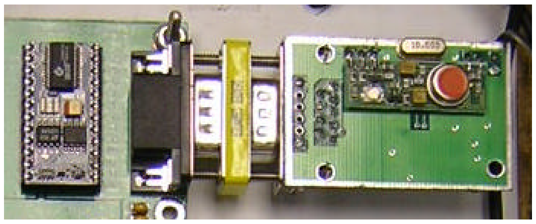


Figure 13. Connection between the micro controller board and the wireless RS232 transmitter board.

Digital Compass

An initial experiment with an analog compass (1525) proved disappointing, showing very erratic readings. We do not recommend the use of this compass. A very effective solution for digital compassing is the HMR3100 from Honeywell [3]. It uses two solid state magnetic sensors and provides very stable readings. Because the motors of the vehicle create a magnetic field, care has to be taken to place the compass far enough so that interference is minimized. We found that a distance of 9 inches from the motors provided stable readings on the compass.

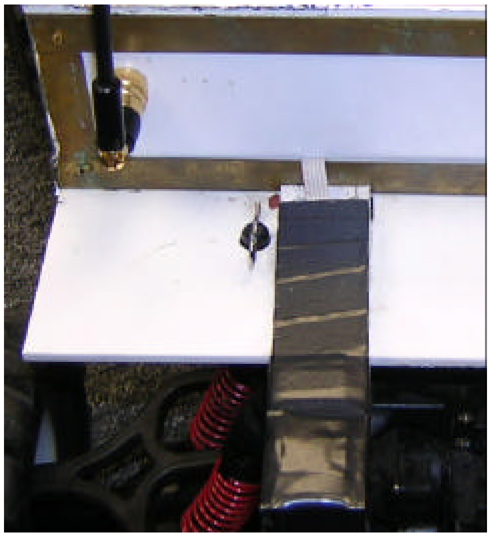


Figure 14. The assembly housing the digital compass.

We built an assembly from styrene pieces to allow the compass to be positioned in the back of the vehicle, about 9 inches from the motors. To shield the compass electronics from the elements the assembly was enclosed with black electric tape (see Figure 14).

Tilt Sensors

The tilt sensors are implemented with a Memsic 2125. It provides dual axis tilt sensing using a thermal accelerometer. The tilt sensor is connected directly to the breadboard area of the micro controller board (see schematics in Appendix 11.4) and Figure 12 above. The tilt is obtained from the following equation: ax = arcsine(x/g), ay=arcsine(y/g) where x,y are the readings from the chip and g the gravitational force. Refer to [5] for more details.

Distance Sensor

The distance sensors are implemented by two Devantech SRF04 Ultrasonic Range Finders [4]. They are attached under the front part of the vehicle box using a right-angle assembly built out of styrene plastic pieces glued together (use cyanocrylate).

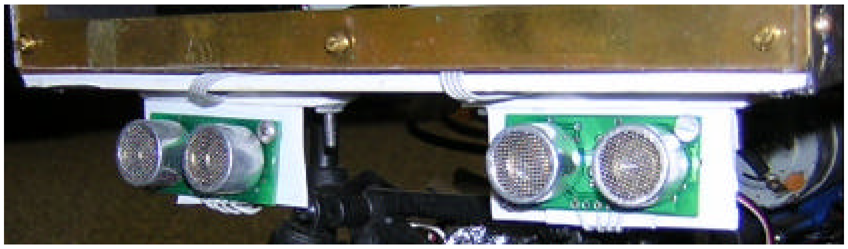


Figure 15. The two ultrasonic range finders.

These range finders have a cone of detection of about 30 degrees. Care has to be taken that they do not point towards the floor so as not to detect grass and minor ground objects. In our experience, a slight upward tilt (5 degrees) decreases the occurrence of false positives. To increase the field of view, the two assemblies are installed at an angle (about 15 degrees).

3.5. Human Driving Interface

In order to capture a user's input actions on the PC workstation when driving the vehicle (training phase) and let the PC drive the vehicle (reinforcement learning phase), the following human driving interface has been designed. The trainer, sitting at the workstation is wearing video goggles which display the video transmitted by the vehicle. He or she can drive the vehicle as one would drive a car, seeing what the vehicle sees, which is much easier than driving by direct sight. The trainer uses an off-the-shelve joystick (for example Logitech Wingman Extreme) to command the vehicle. The joystick is connected to the PC workstation via the game port or alternatively the USB bus. The capture program reads events from the joystick port, stores them into files and passes them back out to the vehicle. No off-the-shelve component exist (to our knowledge) to control remotely such a vehicle from a PC, we built a custom interface. Detailed schematics of this interface can be found in Appendix 11.2. The main component of this interface is a Basic Stamp micro controller board. This is the same micro controller platform as described in Section 3.4 (a BS2SX Basic Stamp together with a Board of Education (BOE)). The micro controller reads data sent to its serial port from the PC and transforms the bytes representing position of the steering and the throttle into a train of pulses that the remote control of the vehicle understands. We use a Futaba 4YF Airplane Remote [8]. The format of that train of pulses is called PPM for Pulse-Position-Modulation. Figure 16 shows an example of such a signal.

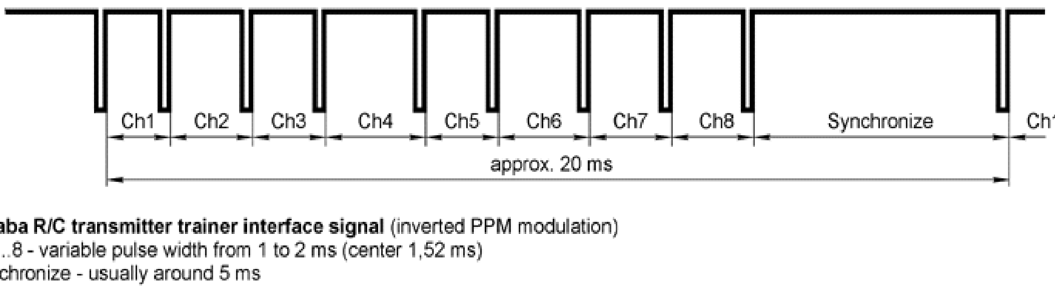

Ch1
Ch2
Ch3
Ch4
Ch5
Ch6
Ch7
Ch8
Synchronize
Ch1
approx. 20 ms
Futaba R/C transmitter trainer interface signal (inverted PPM modulation)
Ch1..8 - variable pulse width from 1 to 2 ms (center 1,52 ms)
Synchronize - usually around 5 ms

Figure 16. PPM train of pulses.

The Basic Stamp is uniquely suited to generate such signals. Its BASIC language has specific a instruction (PULSOUT) to create a pulse of a given length on one of its output pins. The listing of the program can be found in Appendix 11.2. The signal is sent to the remote via a 'trainer' cable which connects to the back of the remote and can be switched on via a toggle switch (see Figure 17).

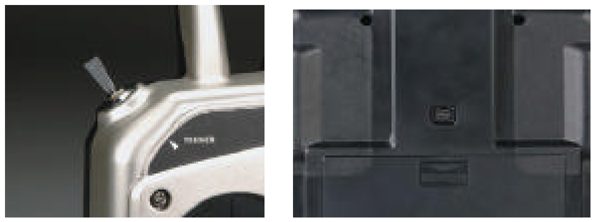


Figure 17. Trainer cord switch and connector on the Futaba 4YF airplane remote.

Note that this remote has evolved from the version we initially used. Instead of the toggle switch, only a push-button was present, making it necessary to actually replace it with a toggle switch. This new revision of the remote makes this step unnecessary. However, the format of the trainer cord connector has changed and hence a new trainer cord needs to be purchased. To connect the trainer cord to the micro controller board, we simply hack it and solder connectors (see Figure 18).

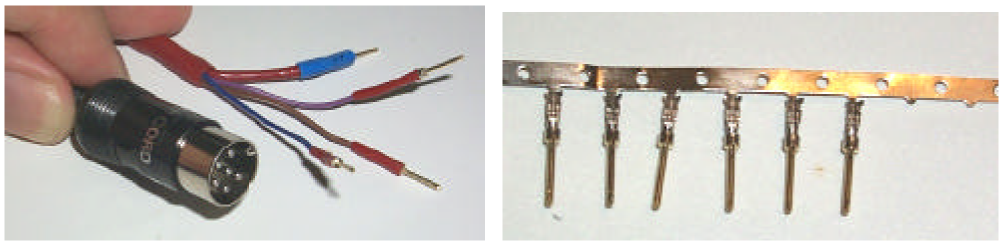


Figure 18. The hacked trainer cord and the connectors used to connect it to the BOE (Radio Shack: 25-Pin Male Crimp-Style D-Sub Connector #276-1429).

The signal needs to be amplified to safely travel the length of the trainer cable. We use a simple amplifier based on a single NPN transistor (see the schematic in Appendix 11.2 for details, and Figure 19).

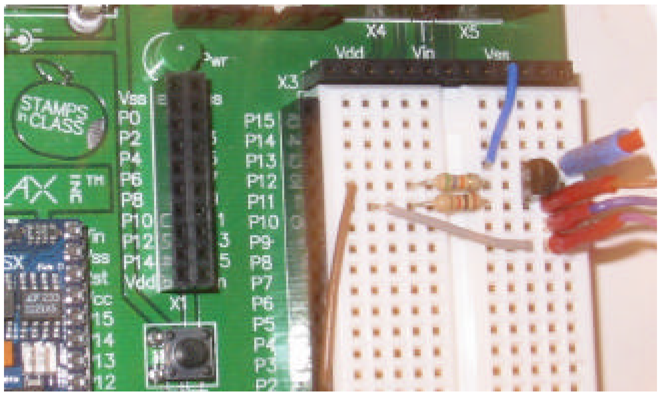


Figure 19. A picture of the micro controller board with the attached trainer cable going to the Futaba remote.

3.6. Wireless Sensors Receiver

The receiver for the sensors is a 418-Mhz Abacom RTcomRs232Rx [6]. It is connected to the PC via a crossed-over serial cable. Additionally, it needs to be powered through the serial cable (which is not standard on PCs). Hence the cable has to be hacked to supply power. Additionally, to increase the range, we used a special antenna (GP-433-BNC) that can be connected to the receiver (soldering needed). The length of the cable between the antenna and the receiver should be kept as short as possible. Figure 20 shows a picture of the box housing both the wireless receiver and the vehicle remote control interface.

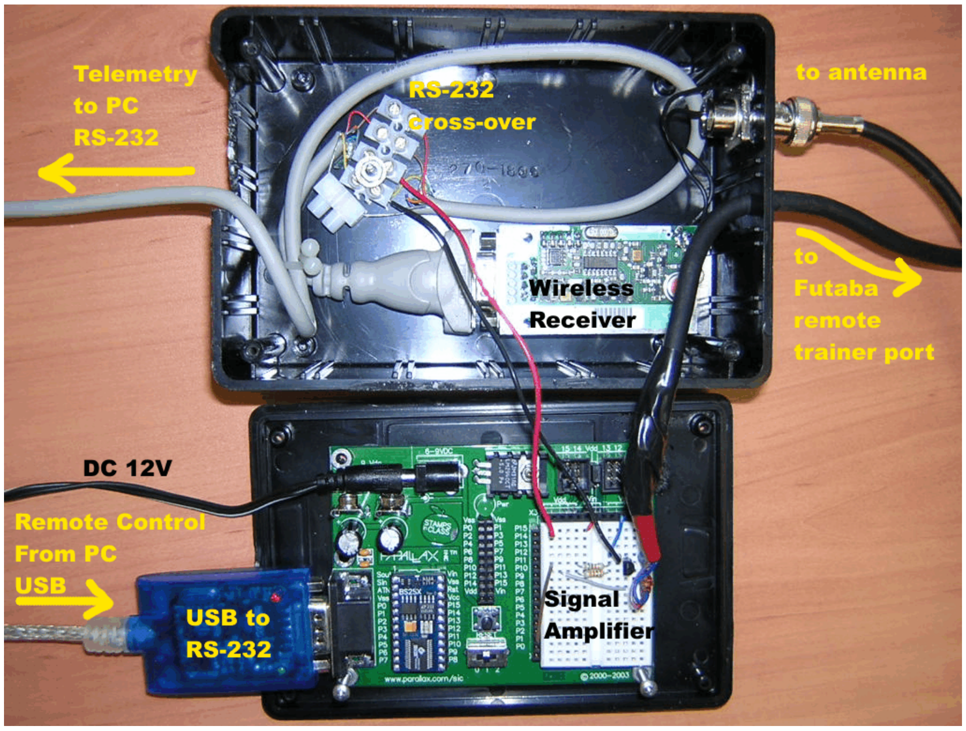


Figure 20. picture of the box that houses both the wireless receiver for the sensors and the PC-to-remote interface.

3.7. Wireless Video Receivers

The video transmitted from the vehicle is received on the PC by two (one for each channel) 900-Mhz video receivers (see Figure 21 and [9]). Each video output is connected to a video capture board [10] (For Linux compliance, a good choice is a video capture card that has the Brooktree BT-848 or BT-878 chipsets, an example of such board is shown on Figure 21).

Figure 21. 4 Channel 900-Mhz video receiver (left) and video capture card (right).

To increase the range and quality of video, we added separate 900-Mhz antennas (see bill of material). The antennas come with male UHF-style connector, which need to be converted to a male F-style connector. To conveniently attach the antenna directly to the receivers, a right-angle connector is needed as shown in Figure 22.

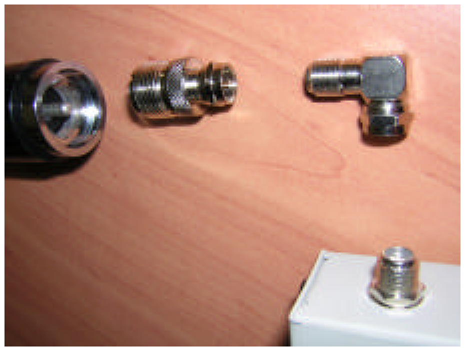


Figure 22. The connectors needed to adapt the UHF-style connector of the 900-Mhz antenna with the F-type input of the 900-Mhz receiver.

3.8. Outdoor Capture Station

To capture interesting data we had the vehicle driven in various outdoor settings. The PC workstation was securely encased inside a moving cart [11] and powered by a generator [12] (see Figure 24). Figure 23 shows a picture of the data capture station. The operator controls the vehicle as well as the recording of the data from the joystick (the trigger is used to start/stop capturing, the twist axis is used for steering and the lever on the left side of the handle for the throttle). Video goggles are connected to the video receivers and provide a glare-free view of the video, even in broad daylight.

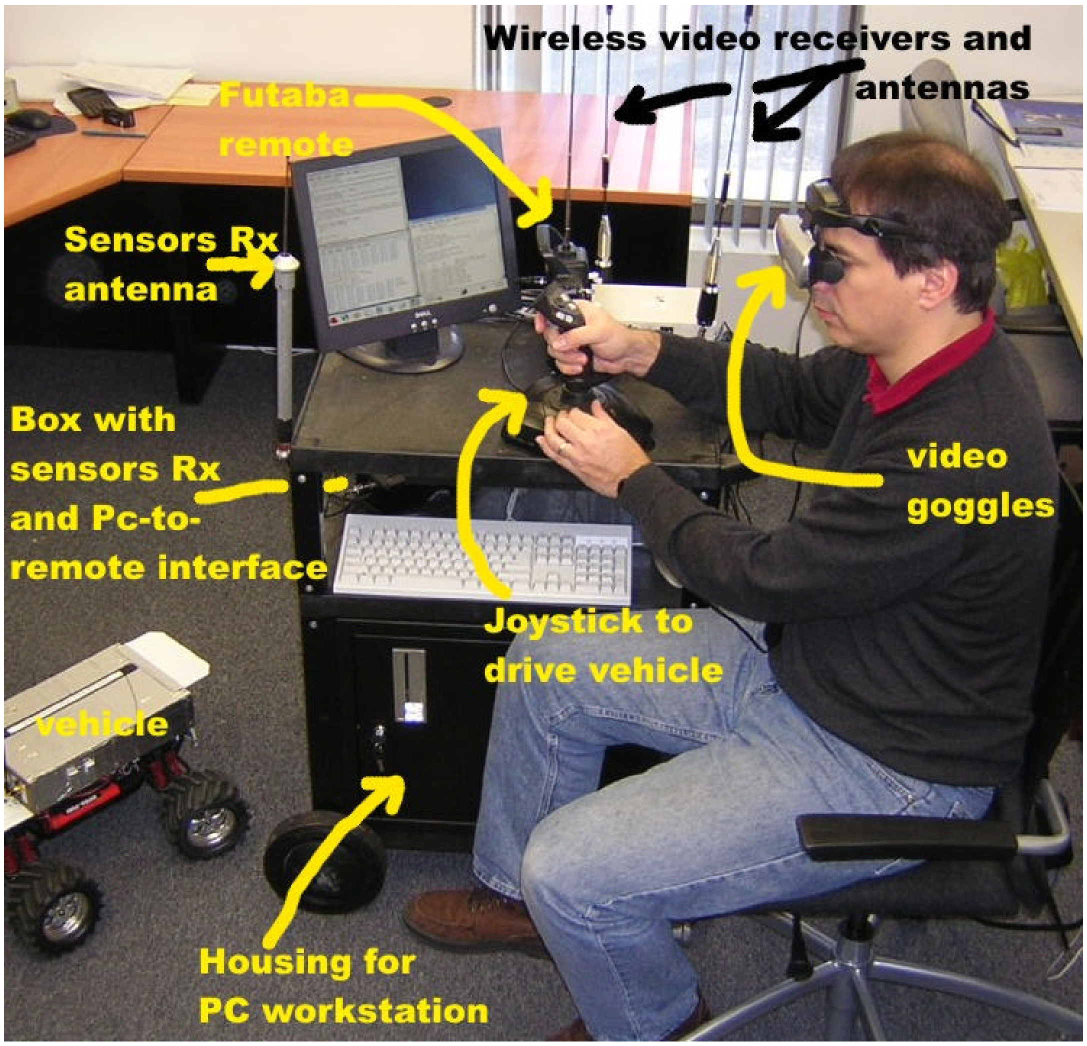

Wireless video receivers and antennas
Futaba remote
Sensors Rx antenna
Box with sensors Rx and Pc-to-remote interface
video goggles
Joystick to drive vehicle
vehicle
Housing for PC workstation

Figure 23. The manned capture station. All components are securely housed within the movable cart. Both the cart and power generator can be easily loaded in and off a minivan and transported to remote locations by a single person.

Figure 24. Honda EU1000iA2 Portable Generator (120 V, 1000 W). This unit was able to operate the computer and charge battery packs for the vehicle at the same time.

3.9. Battery Charging

The vehicle needs two packs of 7.2-V batteries (6-cells). 3000 mAh or higher are recommended in order to increase the length of data capture runs. We used NiMh because they are simpler to maintain (no reconditioning necessary). To efficiently charge these two packs together, we use a digital peak charger from Astroflight (see part list). The charger can be connected directly to the DC output of the power generator or, providing an adequate power supply (see part list) to a 120-V AC outlet for overnight charging. The battery charger's connectors need to be adapted in order to charge both battery packs simultaneously. Figure 25 shows the connectors and cable needed to connect the pack ***in series*** to the charger.

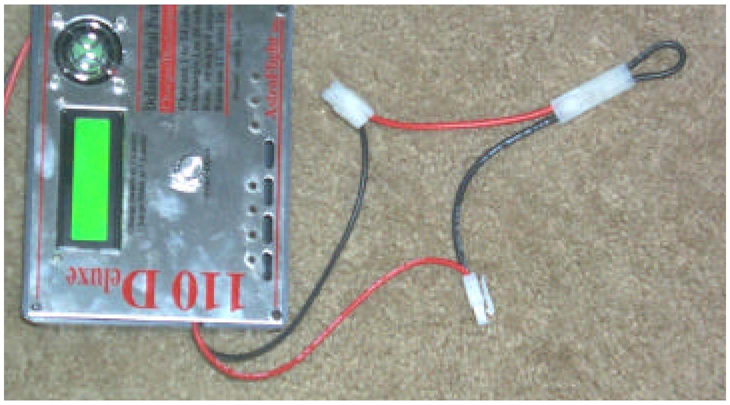


Figure 25. To connect up to three 7.2-V battery packs to the charger, two following is needed: 3 female and 1 male Tamiya 7.2-V connectors, a pin extractor for polarized connector (cat #274-223 at Radio Shack) and 14AWG wire.

4. Machine Learning Environment

4.1. Lush Software

The machine learning environment chosen is called Lush. Lush is an object-oriented programming language designed for researchers, experimenters, and engineers interested in large-scale numerical and graphic applications. Lush is designed to be used in situations where one would want to combine the flexibility of a high-level, loosely-typed interpreted language, with the efficiency of a strongly-typed, natively-compiled language, and with the easy integration of code written in C, C++, or other languages. Furthermore, lush provides interfaces for video capturing boards and the necessary hooks to build interfaces for other custom hardware as will be required for this project.

Lush implementations of the targeted machine learning algorithms are available as free software (under the GPL license) and runs on GNU/Linux, Solaris, and Irix: http://lush.sourceforge.net/.

4.2. Computer Hardware For Network Training

Two separate computers were used for storage of training data and for conducting the network training experiments. The system configuration is shown in Table 1.

Item	Description
Manufacturer	Penguin Computing
Model	Relion 130 Server
CPU	Dual Intel Xeon, 3.06 GHz clock
Memory	512 MB Low-pro DDR PC2100
System hard drive	40 GB Seagate 7200 IDE V2
Data hard drive	200 GB Maxtor 7200 IDE w/8MB
Chassis	1.75” Rack mount
OS	Linux Red Hat 8.0

Table 1. Configuration of the compute servers used for training. Two equally configured servers have been in use.

The two compute servers were configured such that both data hard drives were mounted on the other machine and the system hard drive of server one was backed up by server 2. Otherwise the configuration was identical. It was possible to run training experiment on either compute server and have access to the entire set of training data, even if it was stored on the other server.

In addition to the two compute servers, three 200-GB Maxtor USB hard drives have been used to carry the data between the data collection PC and the two training compute servers and for backup of the collected training data.

4.3. End-to-End Learning Overview

Traditionally, autonomous vehicle control would be broken up into subtasks which would then be individually implemented and hand tuned, for example:

- Image preprocessing such as white balance and noise filtering.
- 3-D image extraction.
- Object detection.
- Path planning.

In the presented end-to-end learning approach all these steps are combined into a single network that is trained solely based on examples. The network is highly structured to allow for optimizing individual processing steps but in no part is any hand optimization or initialization done. The network starts with random values and learns entirely from examples.

The overall sensory-motor process takes the images from the stereo cameras as input and produces throttle position and steering wheel angle as output, using end-to-end learning techniques to map inputs to output.

5. Data Collection

Data collection was done by a human driver who drove the vehicle remotely from the PC with the joystick. The driver could see the image of one video camera (no stereo vision) while driving. During each training run the PC recorded the output of the two video cameras together with the steering wheel position and throttle of the joystick and other sensors. Only the data from the video cameras and steering wheel angle was used for supervised learning. The other sensors were only needed during reinforcement learning.

5.1. Strategy

As this data serves for training of the network, it was extremely important to collect the right type of data, e.g., to collect data from all different situations that we want the vehicle to learn, use a great variety of different training grounds and lighting conditions, and to avoid collecting conflicting data. For example, since we want the vehicle to drive straight whenever no obstacle is in sight we avoided collecting any data where the vehicle turns in such a situation. The general strategy for collecting training data was as follows:

- Collect data from as large a variety of off-road training grounds as possible.
- Collect data with differing lighting conditions, i.e., different weather and different time of day.
- Collect sequences where the vehicle starts driving straight and then turns as an obstacle appears.
- Avoid turning when there are no obstacles in sight.
- Include straight runs with no obstacles and no turns as part of the training set.
- Try to be consistent in the turning behavior, i.e., always turn at approximately the same distance from an obstacle and always turn about the same amount to avoid the obstacle.

Even though great care was taken in collecting the highest quality training data possible, there is a number of imperfections in the training data that could not be avoided:

- Since low cost cameras were used, the white balance of the two was not exactly equal and the brightness of the two stereo images was therefore slightly different.
- The wireless video connection caused dropouts and distortions of some frames. Approximately 5% of the frames were affected. Examples are shown in Figure 29 and Figure 30.
- The cameras are mounted rigid on the vehicle and are exposed to vibration of the vehicle. The resulting video clips are quite jerky and sometimes hard to view even by humans.
- The auto brightness adjustment of the cameras was sometimes slow and not very good. Especially driving into the sunlight caused the foreground to become very dark and objects became hard to recognize.

Despite these difficult conditions, the network managed to learn the task quite well as will be shown later.

5.2. Collected Data

The data was recorded and is archived with the full resolution of $320 \times 240 \times 24$ frames per second. However, not the full available resolution was used for actual training (see Section 6).

Data for supervised learning was collected on 17 days between September and December 2003. A total of 3,228 clips were collected with an average length of about 70 frames each. This results in a total number of about 225,000 frames.

Data for reinforcement learning was collected on 3 days in January 2004. A total of 349 clips were collected with an average length of about 70 frames each, resulting in about 24,000 frames.

5.3. Data Cleaning

Sometimes it was unavoidable to make turns even if there was no obstacle in sight, just to bring the vehicle into position. These parts of the video sequences were later cut out manually. No other manual data cleaning took place.

5.4. Example Images

The following figures show some example images from the collected raw training data including the steering wheel angle and throttle position collected from the human driver. Figure 29 and Figure 30 show examples of problems with the wireless transmission. Similar noise and dropouts occurred in approximately 5% of the frames. It would have been too time consuming to cut them all out manually. They were therefore left in the training data and the network had to deal with them.

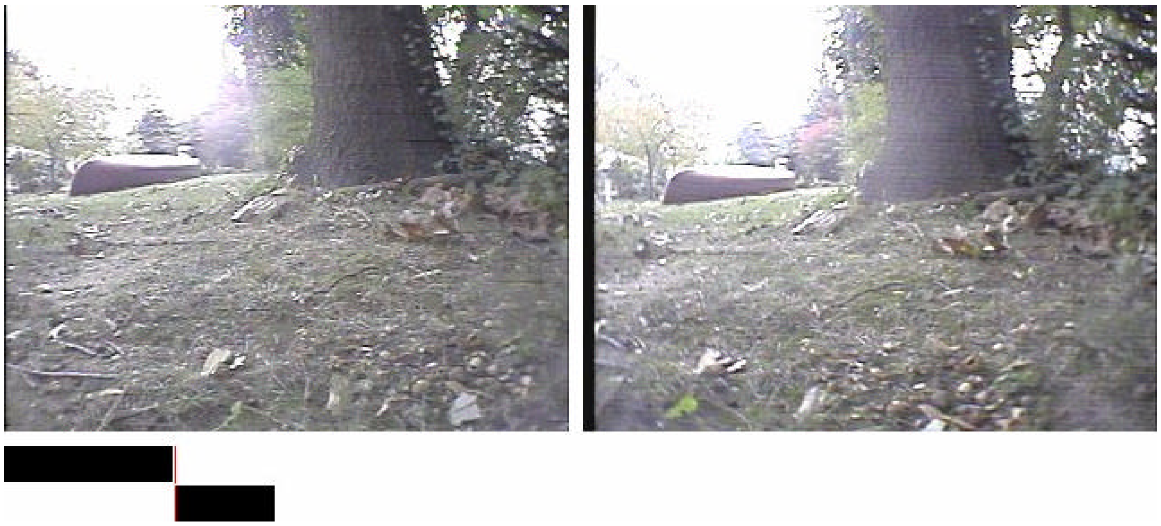


Figure 26. Example video images from the training data. The two pictures are taken by the two cameras at the same time. The two black bars below the images indicate steering wheel angle and throttle position of the human driver.

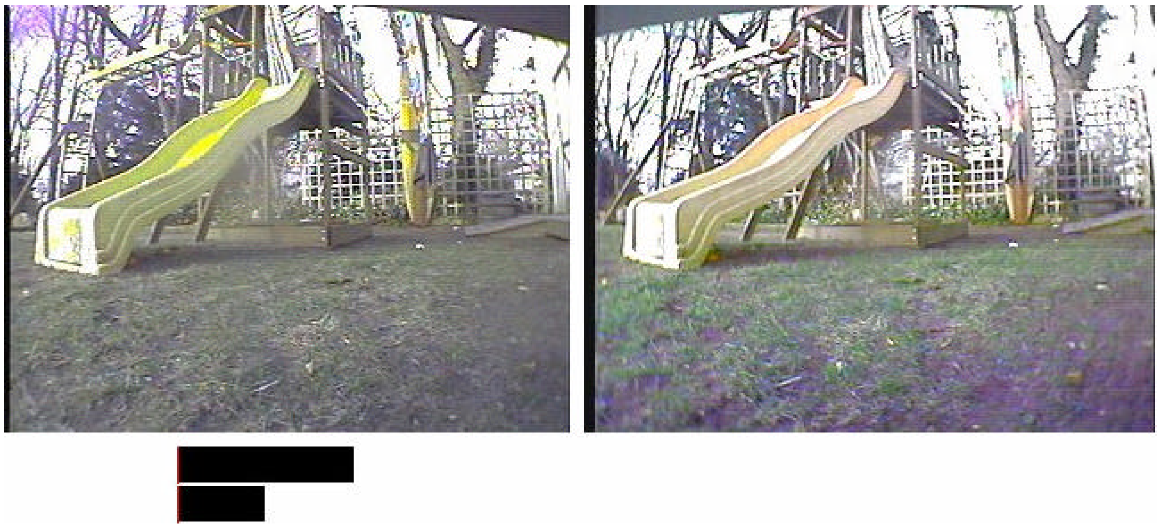


Figure 27. Another example video image from the training data.

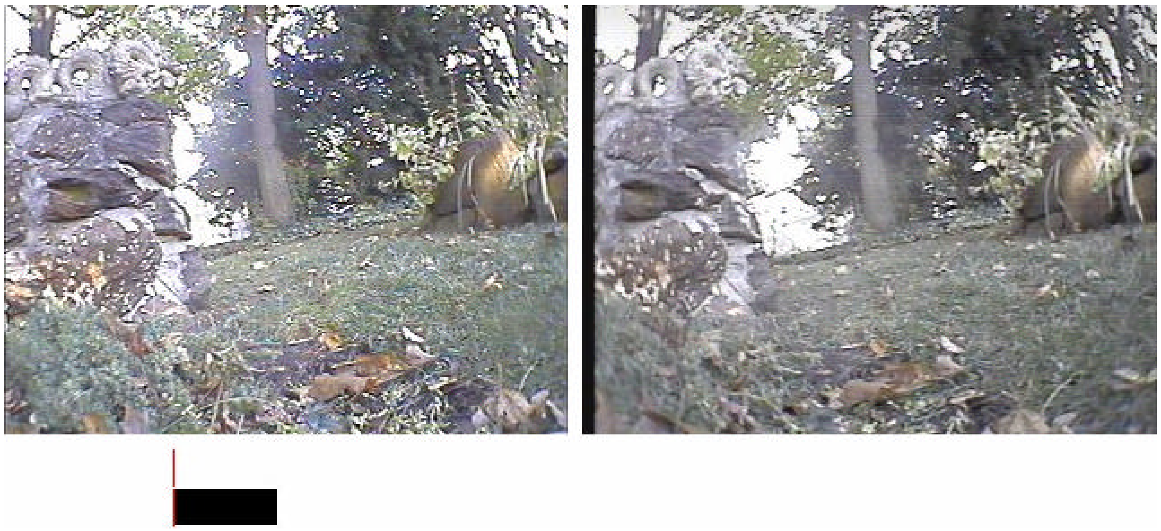


Figure 28. A third example video image from the training data.

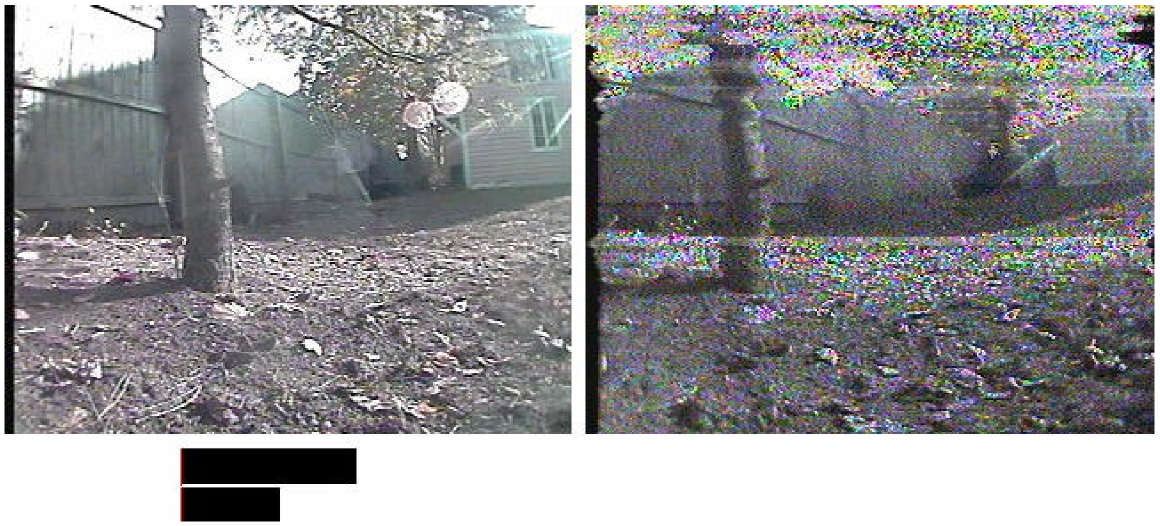


Figure 29. Example video image with poor reception.

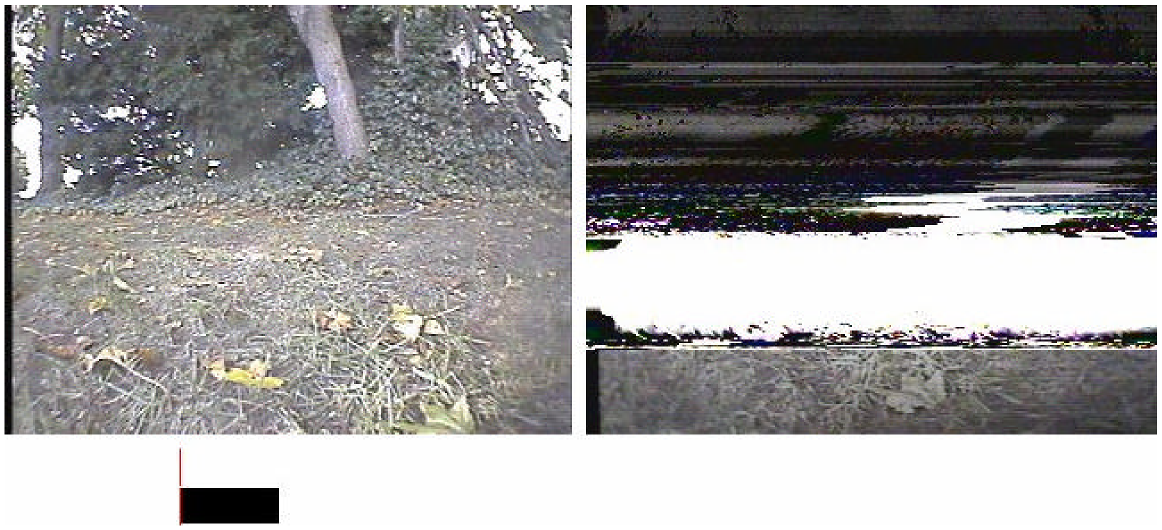


Figure 30. Example video image with a dropout.

6. Supervised Learning

6.1. Introduction

In supervised learning the independent parameters of a network or weights are adjusted in an iterative process to optimize the performance for a given set of training data. The training data contains example input values or input vectors for the network and the corresponding desired output vectors or target vectors. The network output values are compared to the target values and an error is calculated. The function to calculate the errors is called the cost function. It is often the quadratic sum of the differences between output and target values. During learning, they are then adjusted with an iterative optimization algorithm to minimize the output of the cost function. Typically, gradient descent or a modification of it is used. This means the derivative of the cost function to the independent parameters of the network is calculated and the independent parameters are then adjusted by a small amount in the direction in which they reduce the cost function output the most (direction of steepest descent). The process is repeated many times for the entire training data set until an optimum is reached.

The networks used in this project were built from traditional nodes, as illustrated in Figure 31. The output of the node is calculated as the weighted sum of its inputs passed through a nonlinear function of sigmoid shape. It is sometimes called a threshold function because it limits the magnitude of the output.

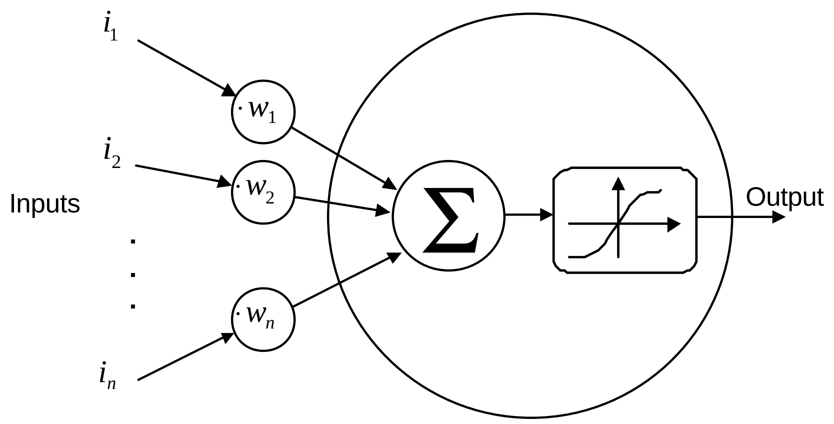


Figure 31. A single network node, as used in this project. The output of the node is calculated as the weighted sum of its inputs passed through a nonlinear function of sigmoid shape. The weights *w* are the independent parameters of the network and are adjusted during training.

Multilayer networks are built by using multiple layers of nodes, as shown in Figure 32. The use of a nonlinear function is a prerequisite for building multilayer networks. Without it the network would be linear and multiple layers of linear functions can always be reduced to a single layer which means the discriminative power of the network would be limited to the one of a single layer network.

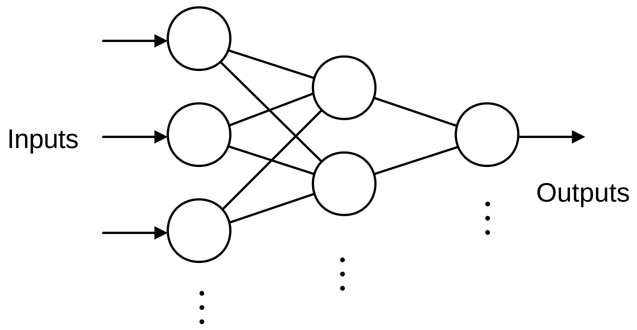


Figure 32. A multilayer network built from individual nodes. The weights are not explicitly shown. They are considered part of the connections between the nodes.

The training of the network is essentially a big numerical optimization process of the weights to minimize the output error, i.e., the output of the cost function.

Once training is complete, the network is tested with a set of data which has not been used during training, i.e., which is unknown to the network. The performance of the network on this test set is what really counts, i.e., the capability of the network to generalize and do meaningful work with data it has never seen before.

A network as shown in Figure 32 is called a fully connected network because the output of each node in one layer is connected to an input of each node in the following layer. By adding layers and increasing layer sizes a network can be made arbitrarily large and it can theoretically learn any task. Practical limits are set by compute power, memory and disk size of available computers but most importantly by the amount and quality of the available training data. The quality of the training data in this context means how well the data represents reality and how diverse it is to cover all aspects of the desired task. This is not necessarily equivalent with, for example, the quality of the video cameras, in our case. What would otherwise be considered poor video quality can still successfully be used by the network as long as it contains the relevant information to perform the task. This is a fundamental difference to traditional image processing algorithms which almost always rely on high quality image sensors.

Each connection in a fully connected network contains one independent weight or independent parameter and as the size of the fully connected network grows, so do the number of independent parameters. The more independent parameters a network has, the more training data is required to achieve good generalization and to avoid that the network simply “memorizes” the training data. In reality, the generalization capability or lack of unlimited training data and compute power limit the use of fully connected networks in many cases. If the network is made too small, on the other hand, it looses the discriminative power to solve the problem and performs poorly, even on the training set. A solution to this problem can be to keep the number of connections in the network high but reduce the number of independent weights by having certain groups of connections share the same weight. How this was done for this particular project is described in the next section.

6.2. Network Design

A convolution network has been chosen whose architecture is inspired by the low-level human nervous system for vision. Such network architectures have previously been successfully applied to handwritten character recognition and object recognition [13]. In such a network the nodes of one layer are connected to a small rectangular area of the layer below, as illustrated in Figure 33. The node to the right of the one shown in layer 2 is connected to the same area in layer 1 but shifted by one or more pixels to the right. This is repeated throughout the entire layer 1 in both dimensions. As an important restriction, the same set of weights is used for all nodes in layer 2. This means that the output of layer 2 is essentially a convolution of layer 1. For example, for a layer one of size 14×14, a convolution kernel of 3×3, and a subsampling rate of 1:1, layer 2 would have 12×12 nodes, 1,296 connections but only 9 independent weights.

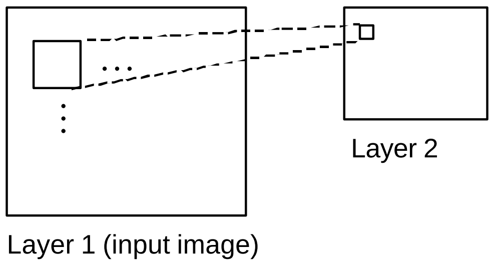


Figure 33. Section of a convolutional network. It shows a single node in layer 2 which is connected to a small rectangular area in layer 1.

A convolution essentially scans for features and layer 2 will become a map of how much the feature defined by the convolution kernel is present at the different locations of layer 1. Multiple plains with different convolution kernels can be used in a single layer to allow the network to look for different features. In any case, the number of independent parameters is very small compared to the number of connections. For image processing applications this network configuration has proven to be a very effective tool to reduce the number of independent parameters while preserving the network's discriminative power.

The actual network used for the supervised learning in this project is organized into six layers with feature maps where feature maps become progressively smaller with each layer. The input layer consists of the raw image data that was sub-sampled to 149×58 pixels. The output consists of two 1×1 feature maps which represent the two driving directions left and right. Figure 34 shows the network architecture.

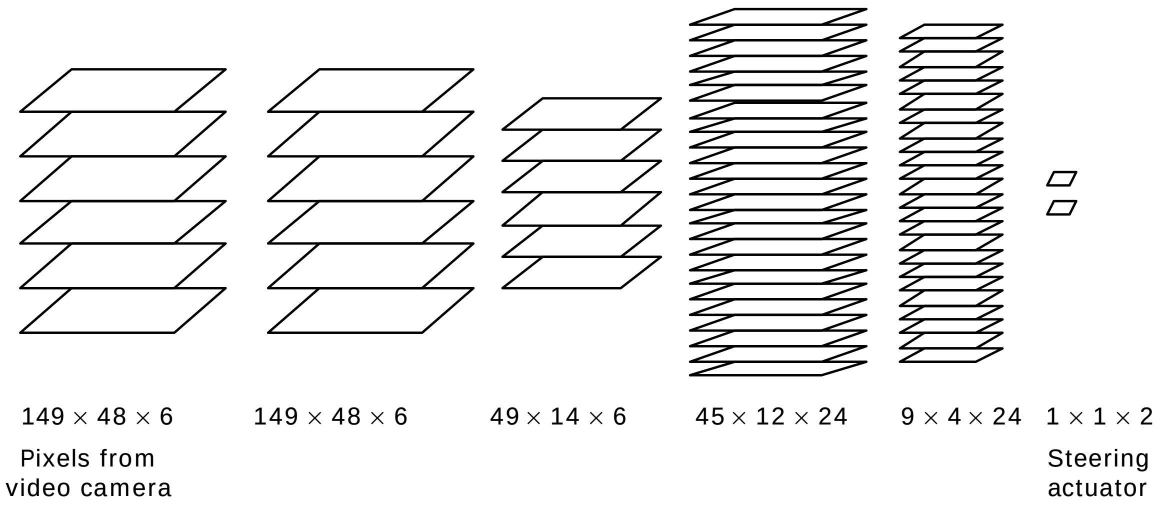


Figure 34. The network architecture for supervised learning. Shown are the feature maps only but not the convolution kernels that connect the feature maps. The input of the network (on the left) is taken directly from the pixels of the video cameras. There are two cameras with three (YUV) color maps each, therefore the six feature maps at the input. The output consists of two values only which are directly interpreted as the steering actuator (left and right).

The first layers (the left two layers in Figure 34) are connected a total of 30 3×3 kernels with a subsampling ratio of 1×1. The next two layers have 8 kernels and a subsampling of 3×4, followed by 96 kernels with 1×1 subsampling, 24 kernels with 5×3 subsampling, 1,920 kernels with 1×1 subsampling and finally 600 kernels with 1×1 subsampling. In total there are 3,148,776 connections but only 71,886 independent parameters.

Out of the available video resolution of $320 \times 240 \times 24$ frames per second, only $149 \times 48 \times 12$ frames per second was being used for training. The reason for this is simply the limitation of compute power. We had to choose a network small enough in size that the PC which controls the vehicle can process 12 frames per second which is considered the minimum frame rate to control the vehicle at the desired driving speed. This limits the total number of connections in the network to roughly three million.

6.3. Learning Experiments Carried Out

In supervised training phase the network was presented with single frames only. Originally we had planned to feed two consecutive frames to the network to allow the extraction of a limited amount of motion information. There is one main problem with this approach, however: when the network sees multiple frames, it's very easy for it to estimate which direction the vehicle is turning. Since the steering values are fairly smooth in time, the network can predict the correct steering angle almost every time simply by looking at where the human driver is turning the vehicle at any given moment. In effect, a good approximation of the desired output is hidden in the input if multiple frames are presented to the network. It was therefore decided to carry out the supervised training phase with single frames only (i.e., single frames in time, the network still had access to the two images from the two cameras to allow for stereo vision).

6.4. Results

Figure 35 and Figure 36 show two snap shots of the trained network in action. The network was presented with scenes that it has never seen before. On top is the subsampled input from the stereo video cameras. The dark blue bar on the left below indicates the network steering output and below are the states of the individual feature maps of the network. The camera input (YUV maps of the two cameras) is on the left and steering output is on the right. This represents the internal state that the network produces when presented with the above image. It is clearly visible that the network was able to learn the task. A better visualization of the results can be seen in the accompanying video clips.

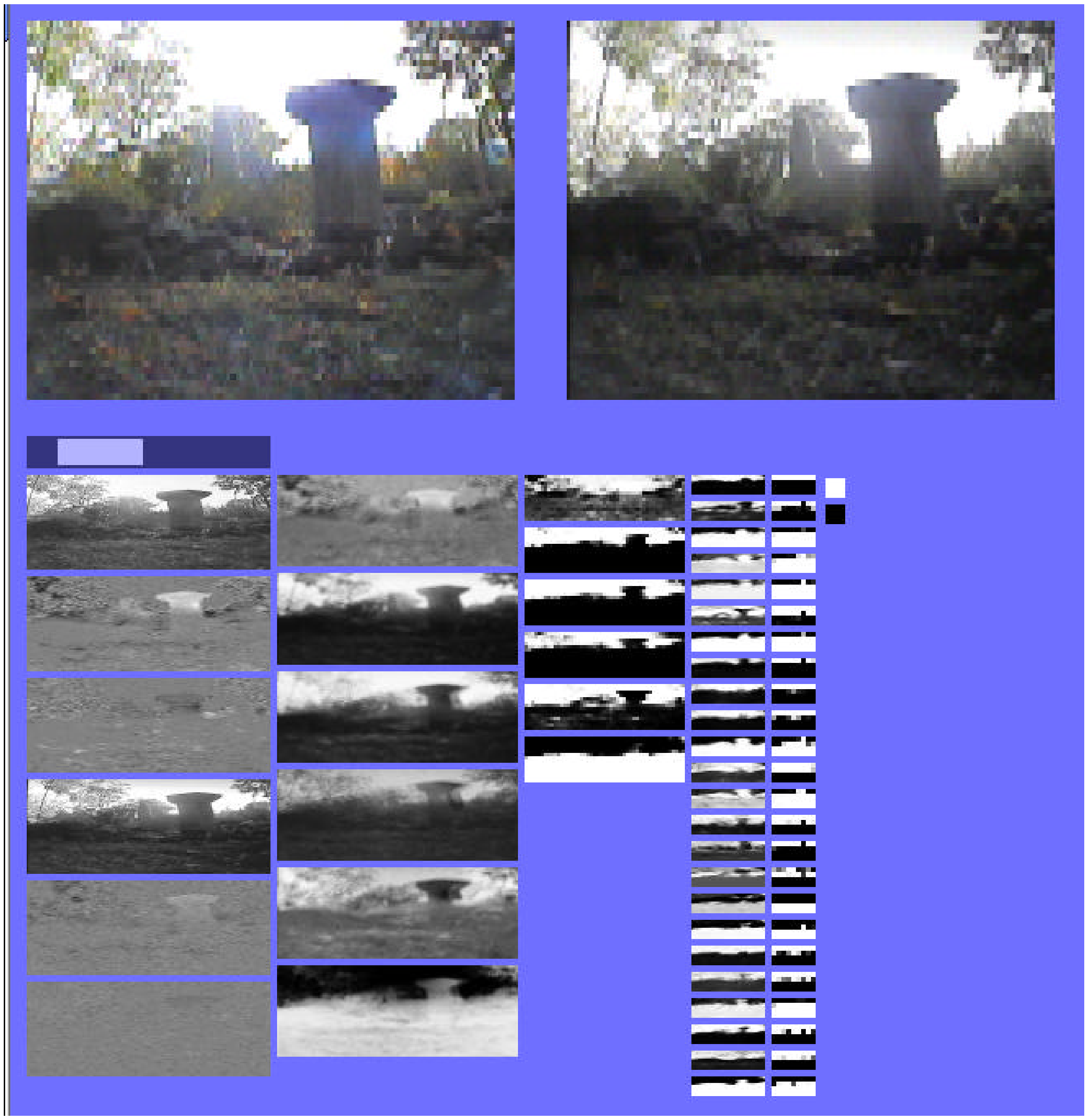


Figure 35. A snap shot of the trained network that was presented with a scene it has never seen before. On top is the sub-sampled input from the stereo video cameras. The dark blue bar on the left below indicates the network steering output and below are the states of the individual feature maps of the network. The camera input (YUV maps of the two cameras) is on the left and steering output is on the right. This represents the internal state that the network produces when presented with the above image.

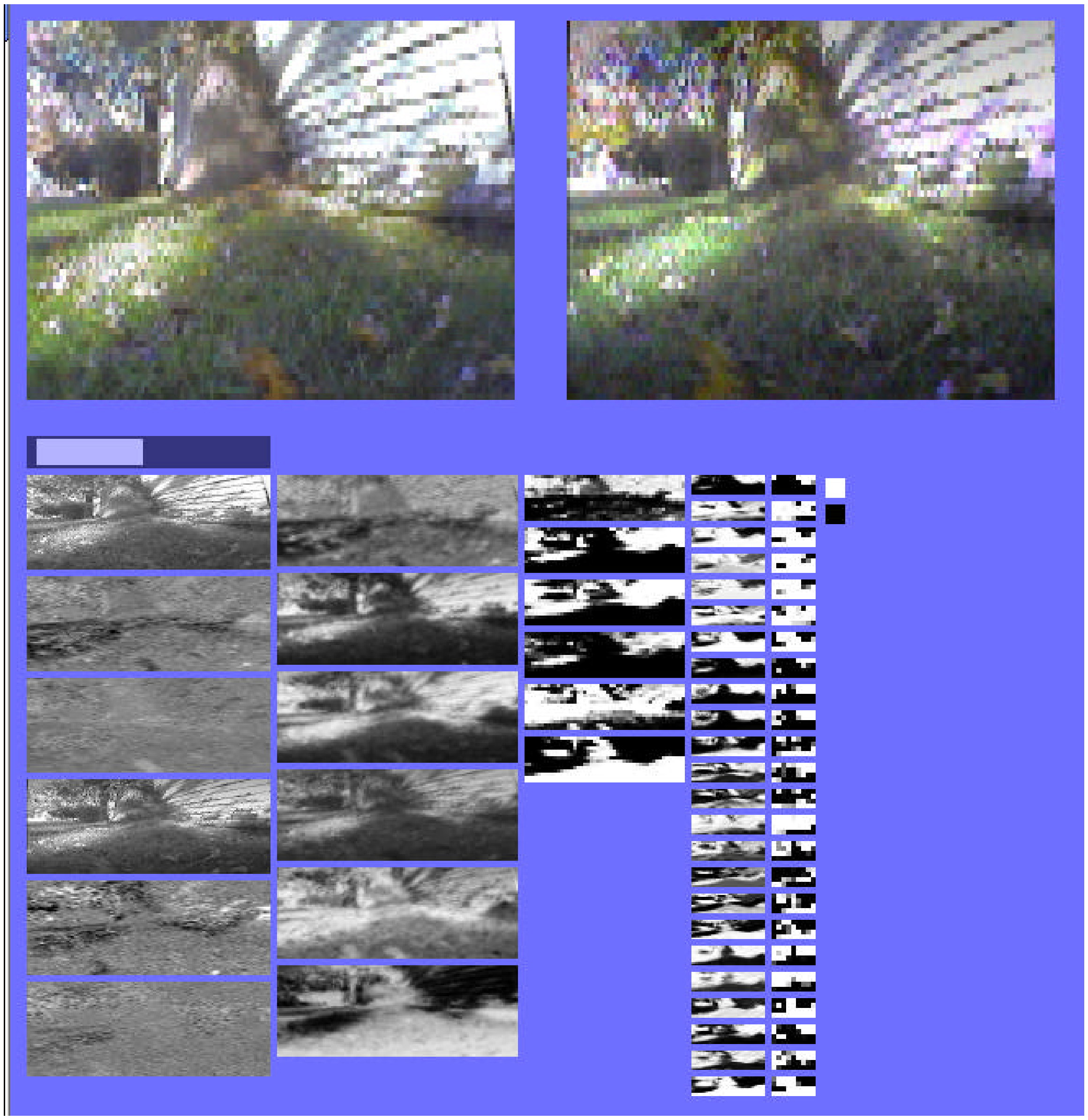


Figure 36. Another snapshot of the trained network.

7. Extended Supervised Learning

This section describes the work which was carried out under amendment P00001 to contract MDA972-03-C-0111.

7.1. Supervised Learning with New Training Data

Throughout several months of collecting training data, the vehicle hardware was continually changed. At the end, considerable improvements have been achieved. These include more robust video transmission with fewer dropouts, adjustments of sonar sensors do avoid noise from the ground and grass, a reliable electronic compass, and a more reliable mechanical transmission of the vehicle which had no more breakdowns and following interruptions of training data collection. Last but not least, the skills of the human vehicle operator (Jan Ben) have been refined.

Training data collected early in the project has therefore less good quality than training data collected later. For this reason it was decided to collected an entirely new set of training data with the latest hardware such that older training data recorded with lower quality hardware can be replaced with higher quality training data. By repeating the supervised training experiments and comparing the results with the corresponding results achieved in earlier experiments with the old training data we can get an idea of the influence of training data quality to the experimental results.

7.1.1. Results

Two experiments have been carried out:

- Starting training with random weights, i.e., without any prior knowledge from previous training.
- Starting training with the best set of weights from the previous training runs (with old training data).

In either case, the network architecture has remained unchanged (as described in Section 6.2). The results are shown in Table 2. The new training data has been divided into a training set (95,421 frames) and a test set (31,800) frames. The frames were taken from approximately 1,500 new video sequences. The training set was used to adjust the weights of the network during training while the test set was not used for adjusting the weights at all. It serves to measure the generalization capability of the network.

After eleven iterations (which took four days of computation time) no significant improvements in the error rate occurred anymore.

Two performance measurements were recorded, the percentage of correctly classified steering wheel angles and the so-called energy. The energy is the Euclidean distance between the output produced by the network and the target output averaged over all input patterns (frames). It is being used to calculate the derivatives for weight update during training, which is essentially a numerical optimization of the weights to minimize the energy.

The percentage of correctly classified steering wheel angles is a measure that cannot be used for training because it is not differentiable. However, it is a more human readable measure to evaluate the network performance. The network output was interpreted by a driver function to be one out of three classes:

- Go straight.
- Turn left.
- Turn right.

The percentage of correctly classified steering wheel angles measures how often the network classified a given input correctly (averaged over all input data). Note, that a classification rate of 74% doesn't mean that the vehicle would crash into obstacles 36% of the time. This is because sometimes two routes around an obstacle are both valid (e.g., go around an obstacle to the left or right are both correct) and the network has chosen one rout but the human driver who recorded the training data has chosen another one. While both are correct, this event will still be recorded as an incorrect classification. Another example is when the network while approaching an obstacle decides to turn a little earlier or later than the human driver.

	Training Set		Test Set	
	Classified Correct	Energy	Classified Correct	Energy
Old training with old data	63.37%	1.2969	42.32%	2.1453
New training with new data starting with random weights	73.95%	0.9046	64.16%	1.2416
New training with new data starting with weights from old training	72.04%	0.9596	63.67%	1.2550

Table 2. Comparison of training results with old and new training data. The training set was used to adjust the weights of the network during training while the test set was not used for adjusting the weights. It serves to measure the generalization capability of the network.

7.1.2. Conclusions

The newly collected training data did improve the network performance significantly as demonstrated with a side-by-side performance comparison of the same network architecture trained with the old data and trained with the new data. We therefore expect that the new training data will serve as a better database for future experiments than the old data does.

7.2. Removal of All 3-D Information for Training

We wanted to find out how well the network performs if all 3-D information is removed. This was achieved by feeding only one of the two vehicle cameras to the network. The network can therefore not extract 3-D information from the video images. Furthermore, since only single frames are presented to the network, the use of optical flow to indirectly extract 3-D information from motion is also not possible. This forces the network to work with elements of feature extraction and scene analysis.

The results are compared to those achieved with earlier supervised training with two cameras.

7.2.1. Results

The network architecture used is exactly the same as described in Section 6.2, except that one camera input was removed and replaced with the other camera input. The network therefore has the exact same structure and size but it "sees" the input of one camera twice instead of seeing the input from the two different cameras.

The results are shown in Figure 37 and Figure 38 and in Table 3. In total, seventeen iterations of the new one-eyed training experiment have been carried out (it takes about half a day per iteration).

The same two performance measurements were recorded, as described in Section 7.1, i.e., the percentage of correctly classified steering wheel angles and the energy.

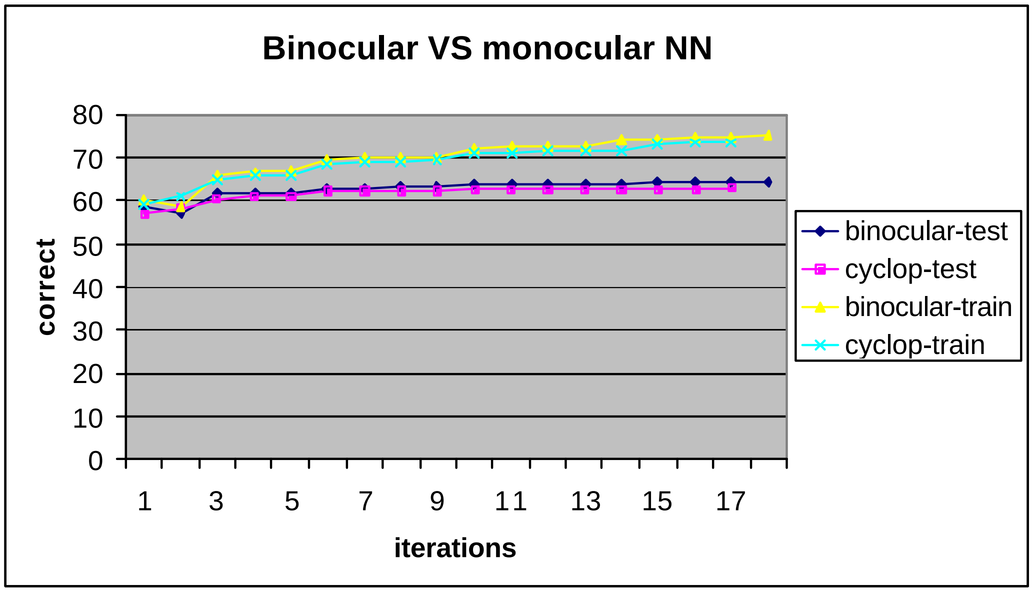


Figure 37. Comparison of classification rate for training and test set between the previous results (binocular) and the new one-eyed experiment (cyclop). The training set was used to adjust the weights of the network during training while the test set was not used for adjusting the weights. It serves to measure the generalization capability of the network. In both experiments, the network was initialized with random weights.

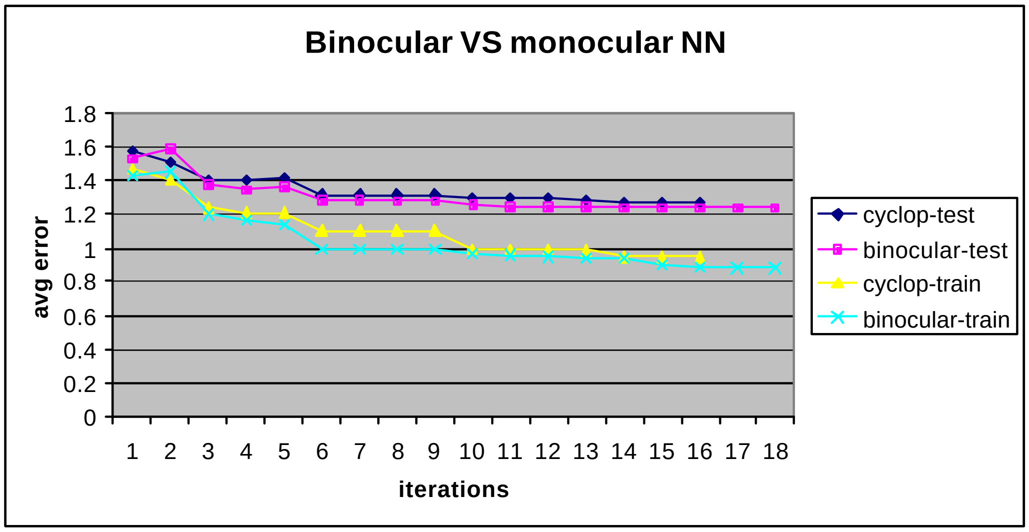


Figure 38. Comparison of energy (mean squared error) for training and test set between the previous results (binocular) and the new one-eyed experiment (cyclop).

	Training Set		Test Set	
	Classified Correct	Energy	Classified Correct	Energy
Binocular: Previous experiment with two video cameras (see also Section 7.1)	74.7%	0.89	64.2%	1.24
Monocular (cyclop): One-eyed experiment (only one video camera fed to the network)	73.3%	0.95	62.8%	1.27

Table 3. Comparison of energy (mean squared error) and classification rate for training and test set between the previous results (binocular) and the new one-eyed experiment (cyclop) after seven iterations, i.e., seven training runs through the entire training set.

7.2.2. Further Examinations

Table 3 shows that the one-eyed network (cyclop) performs almost as good as the binocular one. This raises the question how much 3-D information the binocular network actually extracts and uses, if any. To examine this question further we did two things:

1) Numerous empirical trials with presenting various objects in front of the vehicle under different indoor conditions such as rooms hallways and different lighting. In each trial the trained binocular network was compared to the trained cyclop. In most cases both networks perceptually performed the same but it was possible to construct situations which require 3-D information to make for making obstacle avoidance decisions in which the binocular network performed well but the cyclop network failed. This provides at least empirical proof that the binocular network did learn how to extract 3-D information from the two camera inputs and is using this information if necessary.

2) A third training experiment (besides binocular and cyclop) was set up. This time one network input was connected to one camera and the second network input was connected to the difference of the two cameras. Table 4 shows the results and compares them with the other two networks. The performance of the new differential input network is in between the binocular the and cyclop network. The results fall right into what common sense would suggest: The performance is better than monocular (cyclop) for obvious reasons, and worse than binocular because the only thing the network has access to is the difference between the views, whereas in the binocular mode, it can compute several combinations of the two images.

	Test Set	
	Classified Correct	Energy
Binocular: Previous experiment with two video cameras (see also Section 7.1)	64.1%	1.24
Differential input: One camera fed to one network input and the difference between the two camera images fed to the other network input.	63.6%	1.25
Monocular (cyclop): One-eyed experiment (only one video camera fed to the network)	62.7%	1.27

Table 4. Comparison of network performance (after fifteen training iterations) between two camera inputs, one camera and one differential input and only one camera input.

7.2.3. Conclusions

The performance of the monocular network is surprisingly close to the one of the binocular network. This suggests that the network architecture is well suited to do obstacle avoidance based on scene analysis and image interpretation or at least some form of the two. It further suggests that in the chosen training environment the task of obstacle avoidance is doable without 3-D information. However, this conclusion may not apply to different outdoor environments, for example situations where obstacles are hard to distinguish from their background.

The evidence that the network can operate even without 3-D information is important because 3-D information is only available within limited range (in the order of 30 ft) but cannot be relied upon to do path planning very far ahead of the vehicle.

7.3. Demonstration of Progress During Training

To demonstrate that the network actually learns, i.e., to show the progress of the network during the training phase we captured the weights in three stages: 1) untrained, i.e., random weights, 2) partly trained and 3) fully trained. The partly trained weights were taken from about one quarter of the first training iteration, i.e., after being trained by about one quarter of the training set for one time. One iteration through the entire training set would have brought the network already to a performance too close to the fully trained network and learning progress would have been hard to recognize. The results are captured on video clips which are included on the same CD ROM as this report. The following three figures show snapshots of each of the three stages.

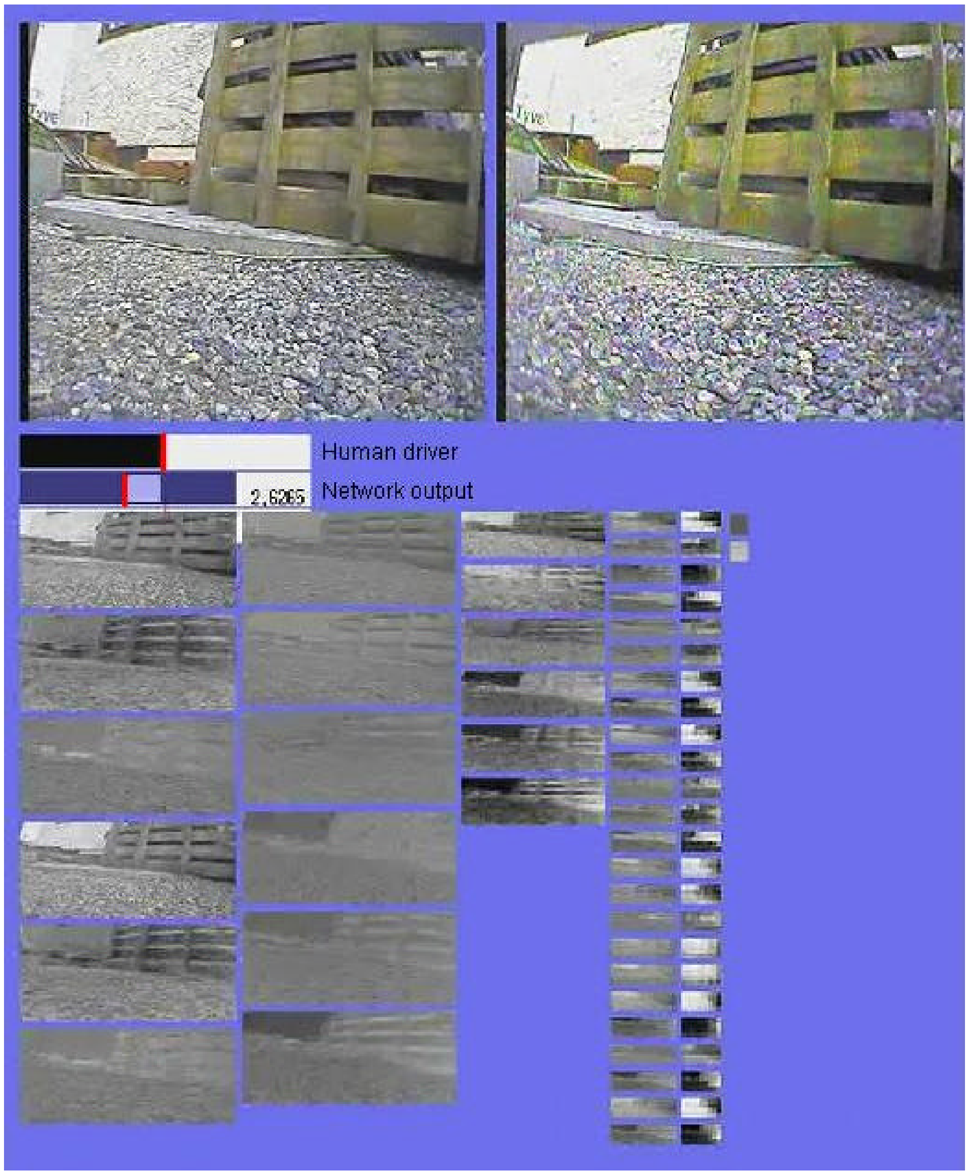


Figure 39. Snapshot of an untrained network with random weights. The snapshot is taken from the test set. The top part shows the two camera input images. The lower part shows the network activation levels. The left most maps are the input images presented to the network (reduced in size and separate frames for the Y, U, V components of the two input images). The center shows the steering wheel angle of the human driver (steering hard left) while the test sequence was recorded. Below it is the steering wheel angle generated by the network (steering slightly to the right). It is evident that the network steers in the wrong direction which is not unexpected since this is an untrained network. Note that the center of steering (red vertical bars) in the picture is not aligned between human driver and network output. The number next to the network steering wheel output is the energy.

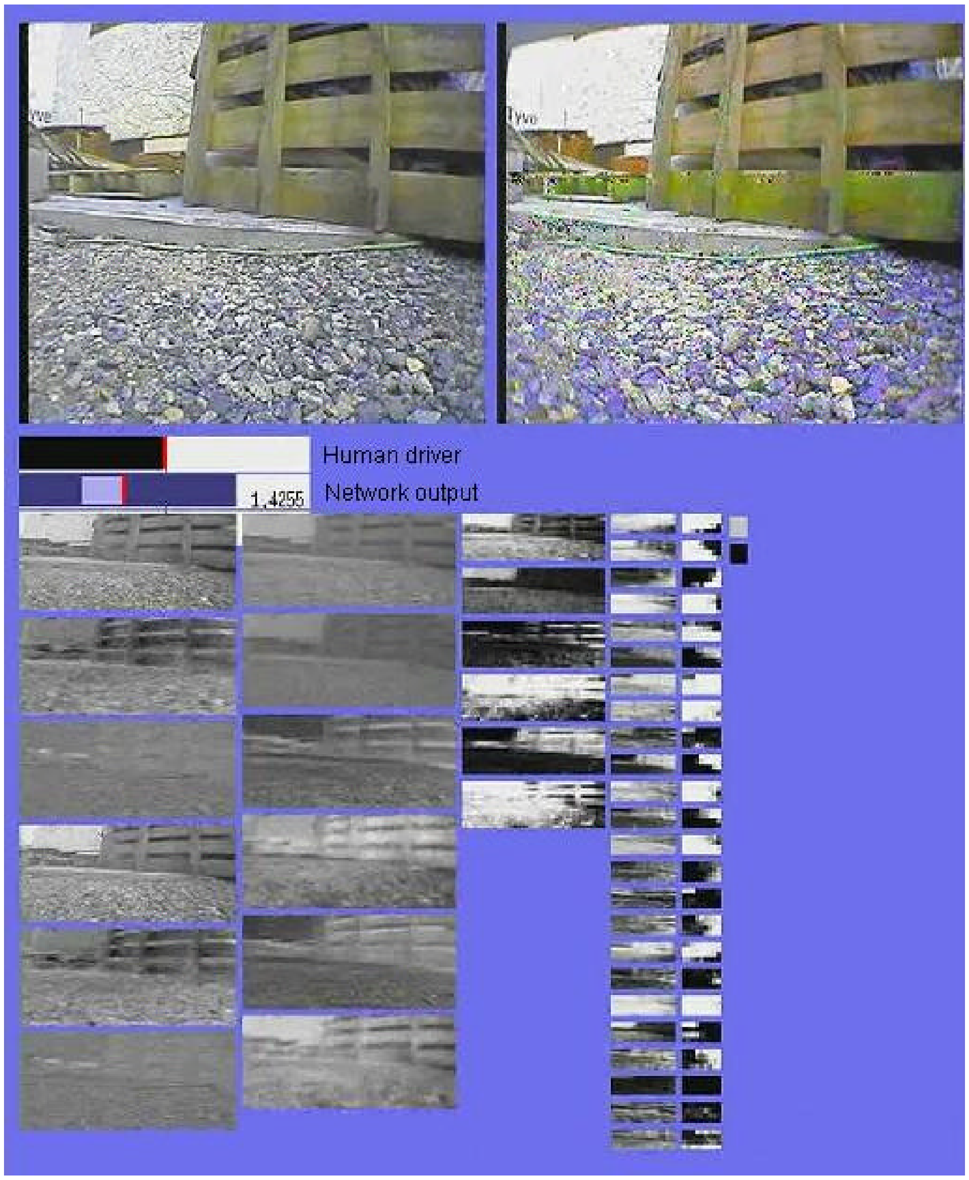


Figure 40. Snapshot of a partly trained network. In contrast to Figure 39 the network now steers slightly to the right which is better than the untrained network. Note the activation levels in the higher level feature maps (to the right) which have sharper images compared to the untrained network.

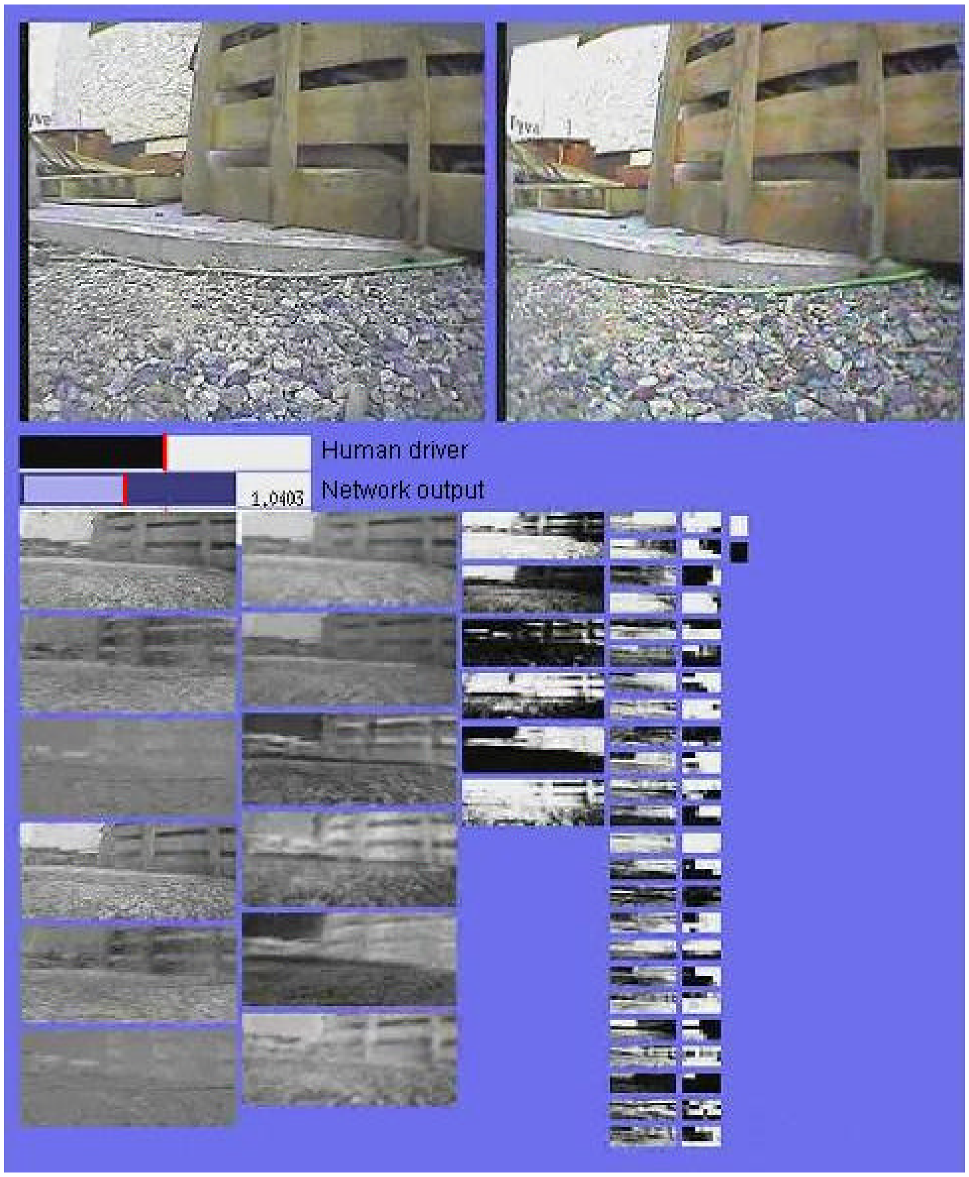


Figure 41. Snapshot of a fully trained network. The network now steers hard left, just like the human driver does. Also, the higher level feature maps show even sharper images compared to Figure 40 which is a sign of the network doing image processing.

8. Reinforcement Learning

8.1. Introduction

The ultimate goal of reinforcement learning is to allow the network to continue learning and improve its performance while driving itself. In other words, it should be capable of learning from its mistakes. For this purpose we have installed additional sensors: two sonar distance sensors, two tilt sensors and an electronic compass. These sensors will let the vehicle determine if it came too close to an object or overturned. Unfortunately, the output of these sensors cannot directly be used as training data for the network because by the time the vehicle overturns or bumps into an object it is already too late. Corrective actions have to be taken several time steps back in the past, as explained in the next paragraph.

In the supervised learning case the human driver effectively gave the network immediate feedback during training. For any given frame of the video sequences the steering wheel angle of the human driver was recorded. The network was simply presented with the (stereo) video image as the input and was asked to mimic what the human driver does, as illustrated in Figure 42.

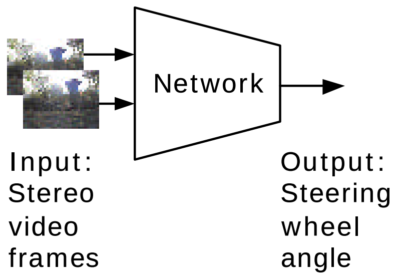


Figure 42. Network configuration used in the supervised learning phase. Inputs are the stereo video frames and output is the steering wheel angle. The cost function during training is based on the error between network output and human driver output.

In case of reinforcement learning, the vehicle has to rely on its own sensors. The interpretation of the sensors is simple: overturning and bumping into obstacles is bad, everything else is good. However, the fact that the vehicle bumped into an object cannot directly be used as training input. The network has to learn how to turn the vehicle before it hits the obstacle but at that time its sensors don't provide any feedback yet.

8.2. Modified Network Design

To deal with this situation the network output has been changed. Instead of outputting the steering wheel angle, the network is trained to predict the value of the various sensors at some time step in the future. In other words, based on the current video image and the current steering wheel angle, the network learns to predict whether or not it will crash or overturn at some time in the near future. This network configuration is illustrated in Figure 43.

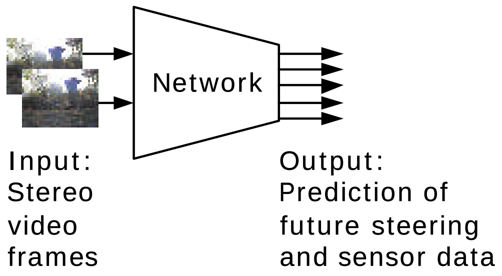

Network
Input: Stereo video frames
Output: Prediction of future steering and sensor data

Figure 43. Network configuration used in the reinforcement learning phase. The input is the same as in the supervised learning phase, i.e., the stereo video frames. The output is different. It predicts the values of the various sensors at some time steps in the future based on the current video input and steering wheel angle.

The network output can no longer directly be used to drive the vehicle. However, once the network learns to predict the near future well, a simple "driver program" can be added at the output of the network which will choose the steering wheel angle that produces the best result in the future, i.e., drive such that the vehicle will not overturn or bump into an object in the future.

The purpose of the electronic compass is to allow the vehicle to stay on an average constant course. However, its data has not yet been used in actual experiments, i.e., up to this point the vehicle has solely been trained to avoid obstacles and drive in a straight line if there are no nearby obstacles.

The network architecture is similar to the one used during the initial supervised learning phase. It is still a convolutional network with six layers. Only the last layer is significantly different from the supervised network. Instead of consisting of only two outputs for steering wheel angle, it now consists of 40 outputs organized in four groups of 10 outputs each (see Figure 44). The 40 network outputs are used as follows:

- 2 outputs to predict the steering wheel angle, one for left and one for right steering.
- 2 outputs to predict the throttle (one for forward and one for backward motion).
- 2 outputs to predict the values of the sonars, one for each sonar.
- 2 outputs to predict the tilt sensors.
- 2 outputs to predict the compass (one for the sine and one for the cosine component).

Four such groups of 10 outputs each were used to predict values in the for time steps of the immediate future. A time step is defined by the capturing frequency of the video frames which is about 8 frames per second. The capturing frequency was limited by the compute power of the PC which controls the vehicle. In contrast to collecting data for the supervised learning phase where the entire CPU was free to capturing frames, the same CPU now had to also drive the vehicle.

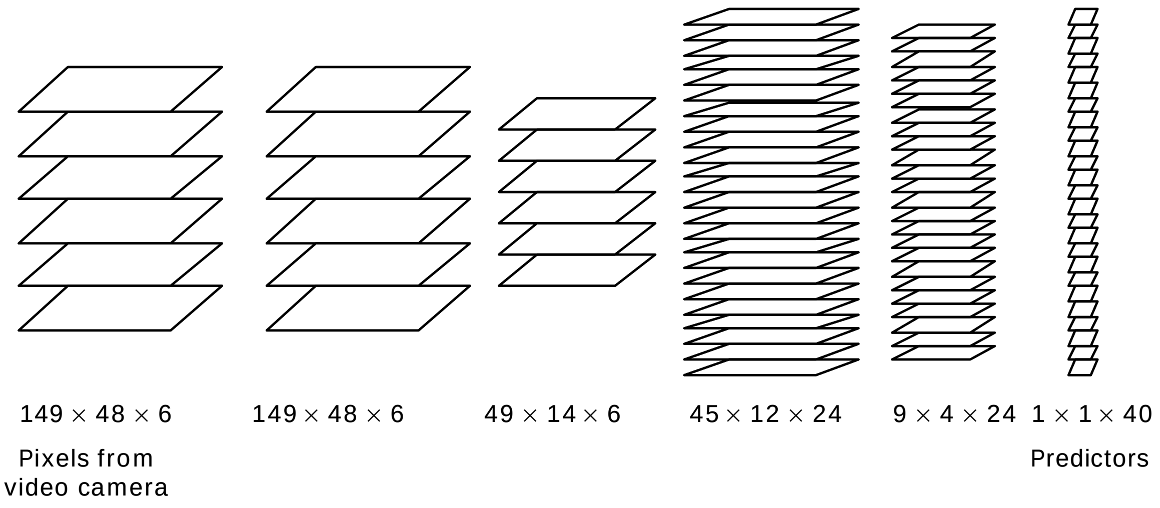


Figure 44. The network architecture for reinforcement learning. Shown are the feature maps only but not the convolution kernels that connect the feature maps. The input of the network (on the left) is taken directly from the pixels of the video cameras. There are two cameras with three (YUV) color maps each, therefore the six feature maps at the input. The output consists of 40 nodes organized in four groups of 10 each. Each group predicts the output of the various sensors one time step further into the future than the previous group.

8.3. Modified Data Collection

In the reinforcement learning phase of the project, data collection continued but this time the vehicle was driven by the previously trained network instead of a human driver. The human supervisor would place the vehicle down on the ground facing some obstacle or series of obstacles and then give control of the vehicle to the network, overwriting control only if the vehicle was clearly taking a wrong path towards an obstacle.

As this data serves for training of the network, it was extremely important to collect the right type of data, e.g., to collect data from all different situations that we want the vehicle to learn, use a great variety of different training grounds and lighting conditions, and to avoid collecting conflicting data. The human supervisor was able to place the vehicle into these different situations to collect an appropriate variety of data, but since the driving was not under human control for much of the time it was necessary to do more driving and additional video editing to obtain training data that would generate improved performance. Poor parts of the video sequences were later cut manually. The general strategy for collecting training data was similar as for collecting data for the previous supervised learning phase:

- Collect data from as large a variety of off-road training grounds as possible.
- Collect data with differing lighting conditions, i.e., different weather and different time of day.
- Collect sequences where the vehicle starts driving straight and then turns as an obstacle appears.
- Include straight runs with no obstacles and no turns as part of the training set.

Data for reinforcement learning was collected on 3 days in January 2004. A total of 349 clips were collected with an average length of about 70 frames each, resulting in about 24,000 frames.

8.4. Learning Experiments Carried Out and Results

Two different training experiments with reinforcement learning have been carried out so far.

8.4.1. First Reinforcement Learning Experiment

The training data that was collected contains data from the network driving itself but also the steering wheel angle from a human. The steering wheel angle of the human was not used to actually drive the vehicle but was just recorded for training, i.e., the network was trained to predict the steering wheel position of the human driver. The errors of the different network output were weighted differently, according to their importance. The values are shown in Table 5.

Sensor	**Error weight coefficient**
Steering wheel for direction the network picked.	1.0
Steering wheel angle for other direction.	0.2
Sonar	1.0
Throttle	0.5
Forward/reverse direction	0.2
Tilt sensors	0.5
Compass	0.1

Table 5. Coefficient for how much errors of the different outputs were weighted during training.

The first *n*-1 layers of the network were initialized with the weights of the trained supervised network. The weights of the last layer were initialized with random values. After training was completed a very simple driver was used to test the result. This driver just used the steering wheel angle the network predicted for the next time step.

Result

The result was disappointing. Essentially, the network kept driving a straight course and almost never made an attempt to avoid an obstacle.

The explanation is expected to be as follows. During data collection for reinforcement learning the human driver let the network run on its own for most of the time. Only if the network made a mistake would the human record a different steering wheel angle than the network. This resulted in the majority of the training data teaching the network to drive straight and almost no training data teaching it otherwise.

8.4.2. Second Reinforcement Learning Experiment

A second reinforcement learning experiment was therefore set up in an attempt to avoid the problem of the first experiment. A cost function was created to predict when the human would interfere with the driving of the vehicle. Another problem encountered in the first experiment is a large number of dropouts of the telemetry data. An additional cost function was introduced to predict the validity of the telemetry data, such that dropouts would not lead to feeding the network contradictory training data. Only data where the human interfered and where the telemetry data was valid was then given any significant cost during learning.

Result

The result was better than the previous experiment but still not as good as the results of the earlier supervised learning experiments. At this point it looks like a number of problems that were encountered during data collection contributed to this result:

- Range of wireless transmission of sensor data is limited to about 50m (150ft) of open terrain. Also, a large body (house, etc.) will prevent proper reception of the sensor data. When collecting new data, the receiving station will be located with greater care to alleviate this issue.
- Dropouts of video frames during wireless transmission (significant improvement was achieved with better receiver antennas but no new training data has been collected yet[1]).
- Windshield got scratched by the camera lenses, resulting in a fuzzy spot in the center of the images (windshield has been replaced and clearance between camera lenses and windshield increased, but no new training data has been collected yet).

9. Conclusions

The network trained with supervised learning has learned the task of obstacle avoidance in a real life setup. It had to deal with drop outs and imperfect input video data. The network is highly structured and the structure was hand crafted. However, none of the over 70,000 free parameters of the network was hand tuned and not even hand initialized (the network was initialized with random values).

The network learned from training examples how to extract and process the essential information from the raw (unprocessed) video input data, such as 3-D image extraction, edge detection, object detection and steering strategy for obstacle avoidance. Particularly impressive is how well the network can cope with the less than perfect real life data it was given.

Reinforcement learning has worked in principle but has not yet lead to an improvement of the earlier network trained with supervised learning. This does not mean that reinforcement learning does not work as expected. However, achieving more meaningful results will require to recollect training data with the improved hardware and to fine tune the methods for dealing with noise in the sensor data.

[1] At the time the improvements were made, snow outside prevented the collection of new training data.

10. References

[1] Dean A. Pomerleau. Knowledge-Based Training of Artificial Neural Networks for Autonomous Robot Driving. In Robot Learning. J. Connell and S. Mahadevan, editors. Kluwer Academic Publishing. 1993.

[2] Todd Jochem and Dean Pomerleau. Vision-Based neural network Road and Intersection Detection. In Intelligent Unmanned Ground Vehicles. Martial H. Hebert, Charles Thorpe, and Anthony Stentz, editors. Kluwer Academic Publishers. 1997.

[3] Parallax Inc, “Memsic 2125 Accelerometer Demo Kit (#28017), January 2003, http://www.parallax.com/dl/docs/prod/acc/MemsicKit.pdf

[4] Parallax Inc, “Devantech Ultrasonic Range Finder (#28015), October 2003, http://www.acroname.com/robotics/parts/R93-SRF04p.pdf

[5] Dao, Ricardo, “Memsic Application Note #AN-00MX-007”, May 2002, http://www.memsic.com/memsic/pdfs/an-00mx-007.pdf

[6] Abacom Technologies, “RTcomTX-RS232, RTcomRx-RS232 Transmitter and Receiver Pair”, http://www.abacom-tech.com/catalog/RTcomTX.PDF

[7] Parallax Inc., “Board of Education, Revision C”, http://www.parallax.com/dl/docs/prod/sic/boeman.pdf

[8] Futaba Corporation of America, “Model 4YF”, http://www.futaba-rc.com/radios/futj40.html

[9] Supercircuits Inc. “900 MHz FM Audio/Video Receiver”, http://www.supercircuits.com/STORE/downloads/instructions/avx900r1.pdf

[10] Hauppauge Computer Works Inc.,”WinTV Go PCI video capture board”, http://www.hauppauge.com/html/wintv_pci_datasheet.htm

[11] H. Wilson Company, “H. Wilson 42’’ Plastic utility cart:, available at http://www.officedepot.com.

[12] American Honda Power Equipment Division, “Hand-held super quiet Generator EU1000”, http://www.hondapowerequipment.com/eu1000.htm

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324. Nov. 1998. 46 pages.

11. Appendix

11.1. Basic Stamp Program Listing for Sensor Reading

```
'{$STAMP BS2SX}
' Author E. Cosatto
' Net-Scale Technologies Inc.

' Pins to the Honeywell HMR3100 Compass
RTS         CON 0       ' Ready to send (OUT)
TXD         CON 1       ' Data (IN)
RXD         CON 2       ' Data (OUT)

' Pins to the Memsic 2125
Xin         CON 6
Yin         CON 7
INPUT Xin
INPUT Yin

' Pins to the Devantech SRF04
Trigger          CON     9
Echo             CON     8
Trigger2         CON     11
Echo2            CON     10
INPUT Echo
OUTPUT Trigger
INPUT Echo2
OUTPUT Trigger2

' temp variables
CO1 var word
CO2 var word

' serial config on BS2SX
'   9600bd, 8bit, noparity, inverted
bdr   var word
bdr=16624

' variables for Devantech SRF04 (works for BS2sx and BS2p)
Trig10          CON     30      ' trigger pulse = 10 uS
ToCm            CON     78


samples     VAR   Nib     ' loop counter
pWidth      VAR   Word    ' pulse width from sensor
rawDist     VAR   Word    ' filtered measurment
cm          VAR   Word    ' conversion to cm
checksum    VAR   Word    ' error correction

xRaw        VAR WORD
yRaw        VAR WORD

tr1         VAR WORD
tr2         VAR WORD
tr3         VAR WORD
```

```
tr4         VAR WORD



' ==========================================================
' Main Program
' read alternatively on both channels of the 1298 and send
' the data back to the PC through the serial port
' ==========================================================

high  RXD          ' Normal mode for reading compass
high  RTS          ' Normal Mode for reading compass
INPUT TXD
high DIO_n         ' Set data pin for first start bit.
LOW Trigger        ' Deactivate Devantech

again:             ' Main loop.

      GOSUB Get_compass             ' Get data from compass

      GOSUB Get_Sonar1              ' take sonar reading
      tr3 = cm

      GOSUB Get_accel               ' memsic reading

      GOSUB Get_Sonar2              ' take sonar reading
      tr4 = cm

      checksum = CO1+CO2+tr3+tr4 + xRaw + yRaw

      ' send data byte by byte
      SEROUT 16, bdr, [0]           ' header (2x byte 0)
      SEROUT 16, bdr, [0]           '
      SEROUT 16, bdr, [tr1.HIGHBYTE]
      SEROUT 16, bdr, [tr1.LOWBYTE]
      SEROUT 16, bdr, [tr2.HIGHBYTE]
      SEROUT 16, bdr, [tr2.LOWBYTE]
      SEROUT 16, bdr, [tr3.HIGHBYTE]
      SEROUT 16, bdr, [tr3.LOWBYTE]
      SEROUT 16, bdr, [tr4.HIGHBYTE]
      SEROUT 16, bdr, [tr4.LOWBYTE]
      SEROUT 16, bdr, [xRaw.HIGHBYTE]
      SEROUT 16, bdr, [xRaw.LOWBYTE]
      SEROUT 16, bdr, [yRaw.HIGHBYTE]
      SEROUT 16, bdr, [yRaw.LOWBYTE]
      SEROUT 16, bdr, [checksum.HIGHBYTE]
      SEROUT 16, bdr, [checksum.LOWBYTE]

goto again ' Endless loop.

' ==========================================================
' Honeywell HMR3100 Compass reading
Get_Compass:
PULSOUT RTS, 1250  ' 1ms pulse to initiate transfer
SERIN TXD, 240, [CO1.LOWBYTE]
SERIN TXD, 240, [CO2.HIGHBYTE]
SERIN TXD, 240, [CO2.LOWBYTE]
```

```
RETURN



' ==========================================================
' Devantech SRF04 Left Sonar reading
Get_Sonar1:
PULSOUT Trigger, Trig10               ' 10 uS trigger pulse
PULSIN Echo, 1, cm                    ' measure pulse
cm = cm + 257 'ensures cm bytes are never 0 (reserved for synch)
RETURN

' Devantech SRF04 Right Sonar reading
Get_Sonar2:
PULSOUT Trigger2, Trig10              ' 10 uS trigger pulse
PULSIN Echo2, 1, cm                   ' measure pulse
cm = cm + 257 ' ensures cm bytes are never 0 (reserved for synch)
RETURN

' ==========================================================
' Memsic accelerometer reading
Get_accel:
PULSIN      Xin, 1, xRaw
PULSIN      Yin, 1, yRaw
xRaw = xRaw + 257
yRaw = yRaw + 257
RETURN
```

11.2. Basic Stamp Program Listing for PC-to-remote interface

```
'{$STAMP BS2SX}
' Author E. Cosatto
' Net-Scale Technologies Inc.
'
' creates a PPM signal for use by remote control
' through trainer cord based on data read on
' serial port. Two bytes are read from serial port
' and send to channel 1 and 2 (first and 2nd pulse).
' A total of 6 pulses is sent (compatible with 6 channel
' receivers). The PPM format is the following. Starting
' low, a first pulse of at least 10ms length is sent for
' synchronization. The following 6 pulses have a width varying
' from 1ms to 2ms. These values have to be calibrated with
' the receiver and servos.

tmp   VAR BYTE
bdr   VAR BYTE
ib0   VAR BYTE
ib1   VAR BYTE
ib00  VAR BYTE
ib11  VAR BYTE
gearb VAR BYTE
p0    VAR WORD
p1    VAR WORD
gear  VAR WORD
POL2  VAR BIT

' set initial values to midpoint, in case there is an error
' right away (so we don't send crap to the servos)
p0 = 1486
p1 = 1486

' serial config on BS2SX
'   9600bd, 8bit, noparity, inverted
bdr=16624

' set PIN 2 as output:
' set initial polarity: 0 of noninverted, 1 if inverted
' pin might be inverted if using an inverting amplifier on the pin
DIR2=1
POL2=1
OUT2=POL2

' this has been fixed in the truck to low gear
gear = 0
gear = 1900 - 700*gear

loop:
      TOGGLE 2            ' start synchronization pulse
      SEROUT 16, bdr, [0] ' send synchro byte 0 to PC

      ' read joystick position (2 axis)
      ' times out after 40 millisec to avoid a too long synch pulse
      SERIN 16, bdr, 40, timeout, [ib0, ib1, ib00, ib11]

      ' read twice each byte to correct transmission errors
      ' additional check for value 0 and 255 (never transmitted)
      IF ib0 <> ib00 THEN serror
      IF ib1 <> ib11 THEN serror
      IF ib0 = 255 THEN serror
      IF ib1 = 255 THEN serror
      IF ib0 = 0 THEN serror
      IF ib1 = 0 THEN serror

      ' convert range [1..254] to [1250..2500] for PULSOUT command
      ' this is calibrated to E-MAXX steering
      p0 = ib0
      p0 = 5*(p0-128) + 1500

      ' convert range [1..254] to [1759..2012] for PULSOUT command
      ' this is calibrated to E-MAXX throttle
      p1 = ib1
      p1 = p1 + 1758
      GOTO timeout

serror:
      ' compensate for above instructions in case of error
      ' avoids too much jitter
      PAUSE 20

timeout:
      OUT2=POL2  ' return pin to low for 0.3 ms
      OUT2=POL2
      GOSUB send_pulses
```

```
GOTO loop        ' main loop


send_pulses:

      PULSOUT 2, p0    ' first pulse: steering
      OUT2=POL2        ' return for .3 milliseconds
      OUT2=POL2
      PULSOUT 2, p1    ' second pulse: throttle
      OUT2=POL2
      OUT2=POL2
      PULSOUT 2, gear  ' third pulse: gear (set to low gear)
      OUT2=POL2
      OUT2=POL2
      PULSOUT 2, 1486  ' the rest is unused by the truck
      OUT2=POL2
      OUT2=POL2
      PULSOUT 2, 1486
      OUT2=POL2
      OUT2=POL2
      PULSOUT 2, 1486
      OUT2=POL2
      OUT2=POL2
RETURN
```

11.3. Part Listing

Bill of material for darpa vehicle:

Complete truck and box (except telemetry)

Item Description	Order info	Part #	Qty.	Unit price	Total price
Traxxas E-Maxx R/C truck	TH	TRAC64A5	1	319.99	319.99
Battery Pack, NIMH,3000mAh, 7.2V	TH	DTXC2097	2	36.99	73.98
Pinion 12T	TH	TRAC3942	2	3.09	6.18
Spur Gear, 72T	TH	TRAC4472	1	2.89	2.89
Camera, 1/3'' CCD	SC	PC171XS	2	79.95	159.90
Lens 2.5mm Micro Lens	SC	ML2_5MM	2	29.95	59.90
500mW 900MHz TX ONLY (special order by phone, with RX can be ordered on the web (+69.95)	MC		2	120	240
LM317T volt regulator	RS	2761778	2	1.99	3.98
Toggle switch	RS	?	1	2	2
Terminal Strip	RS	2740680	2	2.49	4.98
Mini Breadboard	RS	2760175	1	7.99	7.99
PCB Terminals (pkg of 4)	RS	2761388	2	2.29	4.58
Snap Plug connector	TH	LXD170	2	1.79	3.58
Battery Pack for onboard electronics	TH	LXMM41	1	15.99	15.99
14 AWG wires	TH	LXJX70	1	3.79	3.79
Styrene sheet	TH	LXDJP6	4	6.99	27.96
Clear Lexan sheet	TH	LXDYW7	1	18.39	18.39
Brass angles	TH	LXRW92	1	17.99	17.99
Brass 2-56 screws	TH	LXKS20	5	1.70	8.50

Brass 2-56 hex nuts	TH	LXKS62	5	1.70	8.50
Flat head 4-40, ½'' screws	TH	LXM819	4	1.90	7.60
Velcro strips	TH	LXD648	1	3.49	3.49

PC-side components

Item Description	Order info	Part #	Qty.	Unit price	Total price
Futaba 4YF remote	TH	LXEFJ4	1	129.99	129.99
Futaba Trainer cord	TH	LXBEK7	1	11.99	11.99
Astroflight battery charger	TH	ASTP0131	1	129.95	129.95
Connectors (Tamiya 7.2V)	TH	LXD170	3	1.79	5.37
Crimp-Style D-SUB connector	RS	276-1429	1	2.49	2.49
Heat Shrink tubing	RS	278-1610	1	2.49	2.49
900Mhz Receiver	SC	AVX900R1	2	99.95	199.90
RFLMA9 900MHz 7dBi Mobile Omni-Directional Antenna	RFL	RFLMA9-7	2	21.95	43.90
Joystick (Logitech Wingman)	BB	3737059	1	39.99	39.99
Parallax Basic Stamp 2sx	PX	BS2SX-IC	1	59	59
Parallax Board of Education	PX	28150	1	65	65
Box for control interface	RS		1	6.34	3.34

Telemetry (on truck)

Item Description	Order info	Part #	Qty.	Unit price	Total price
Parallax Basic Stamp 2sx	PX	BS2SX-IC	1	59	59
Parallax Board of Education	PX	28150	1	65	65
Abacom wireless RS232 Transmitter (418Mhz)	ABA	RTcomTX-418-RS232	1	87.15	87.15
Male-to-Male DB9 gender adapter	RS	26-231	1	6.49	6.49
Devantech SRF04 Ultrasonic Range Finder	PX	28015	2	36	72
Memsic 2125 Dual-axis Accelerometer	PX	28017	1	25	25
Honeywell HMR3100 Digital compass	HW	HMR3100	1	89	89

Telemetry (PC side)

Abacom wireless RS232 Receiver (418Mhz)	ABA	RTcomRX-418-RS232	1	105.51	105.51
433Mhz Receiver Antenna.	ABA	GP433BNC	1	?,<100	
Male-Female DB9 Serial cable	RS	26-117B	1	14.49	14.19
USB to RS232 adapter (if extra serial port needed)	UW	US232	1	29.99	29.99

Miscellaneous (PC side)

Power Generator Honda EU1000	AE		1	659	659
H. Wilson 42" UL-Rated (20" TV)	OD	476632	1	249.99	249.99

Plastic Utility Carts With Locking Cabinet And Big Wheel Kit					

Key to Vendors

ABA: www.abacom-tech.com
AE: American Equipment: www.americanequipmentusa.com
BB: Best Buy (local store)
CC : Circuit City (local store)
HM: Hobby Masters, Red Bank (local store)
HW: http://shop.ssec.honeywell.com
IBX: www.ibexpc.com
IM: www.imagesco.com
LNX: www.lynxmotion.com
MC: www.microcameras.com
PLX: www.parallax.com
OD: www.officedepot.com
RFL: RF Linx Corp., http://www.rflinx.com
RS: radio shack (local store) www.radioshack.com
SC: www.supercircuits.com
TH: www.towerhobbies.com
UW: www.usbwholesale.com
X10: www.x10.com

11.4. Schematics For Vehicle On-Board Electronics

See following pages.

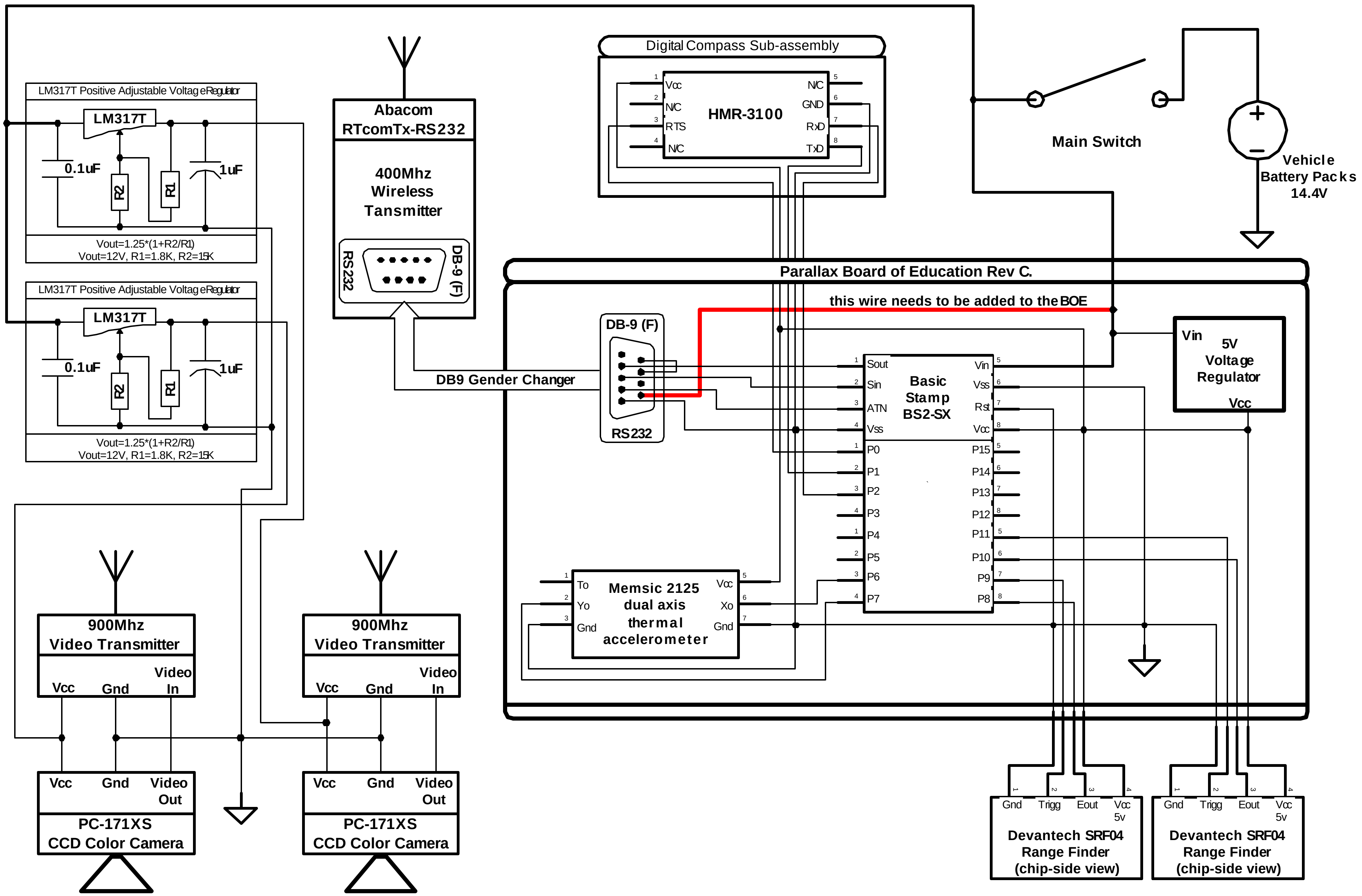
11.5. Schematics For PC-To-Remote Interface

See following pages.

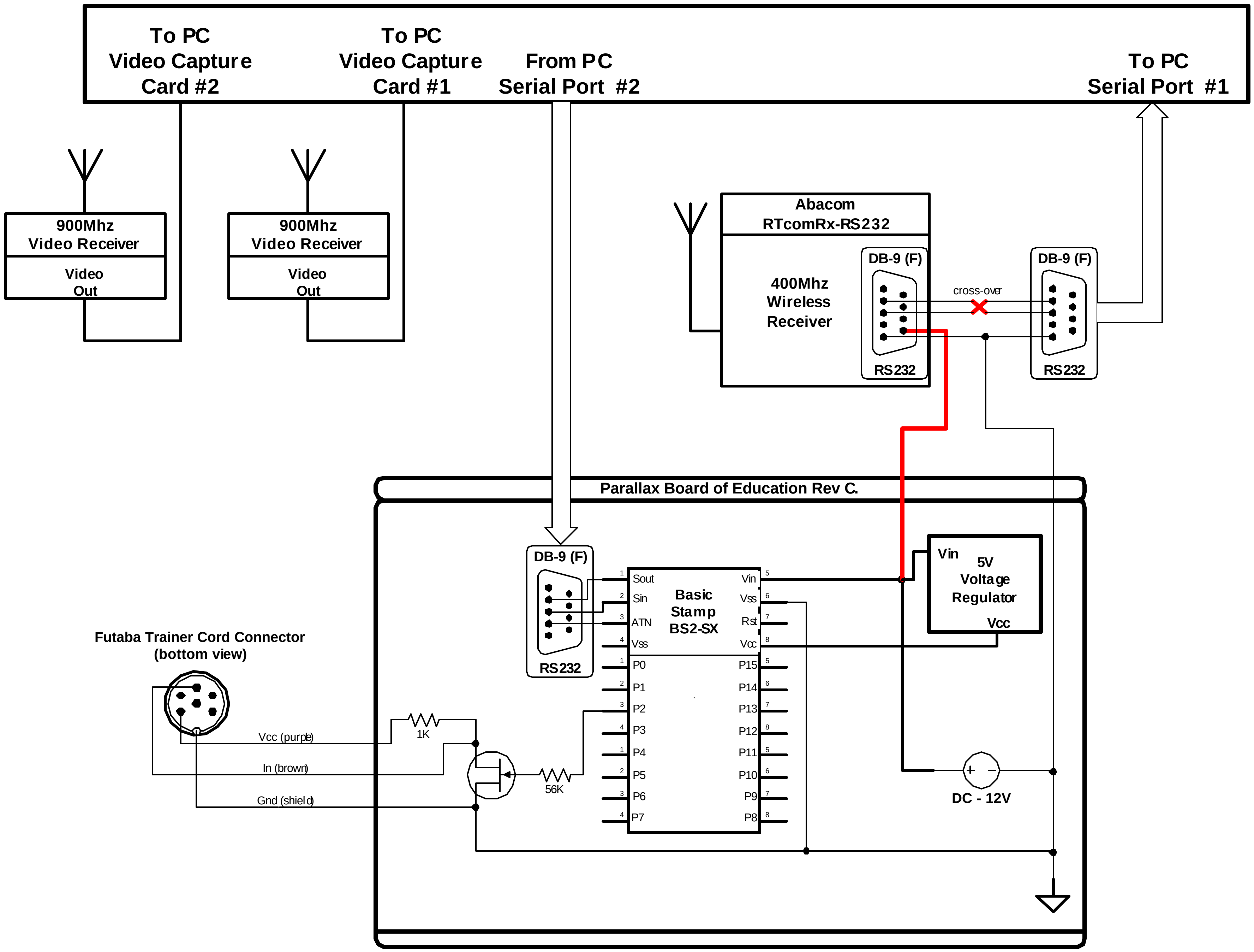
11.6. Schematics for Camera Mount

See following pages.

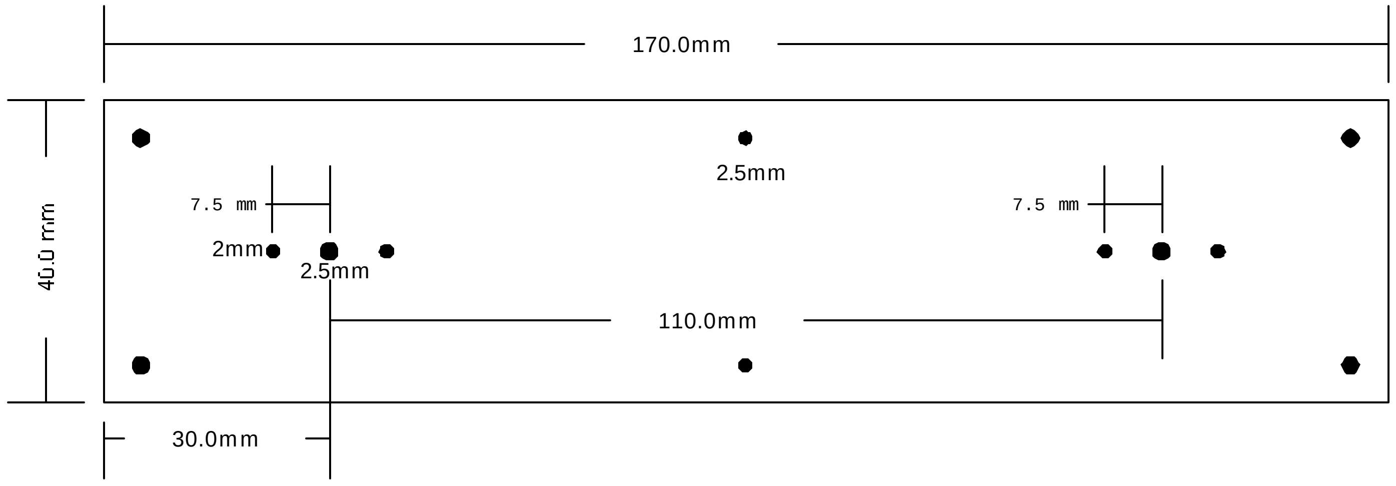
D.A.V.E. On-board Cameras and Sensors


LM317T Positive Adjustable Voltage Regulator
LM317T
0.1uF
1uF
R2
R1
Vout=1.25*(1+R2/R1)
Vout=12V, R1=1.8K, R2=15K
LM317T Positive Adjustable Voltage Regulator
LM317T
0.1uF
1uF
R2
R1
Vout=1.25*(1+R2/R1)
Vout=12V, R1=1.8K, R2=15K
Abacom RTcomTx-RS232
400Mhz Wireless Tansmitter
RS232
DB-9 (F)
DB9 Gender Changer
Digital Compass Sub-assembly
HMR-3100
Vcc
N/C
RTS
N/C
N/C
GND
RxD
TxD
Main Switch
Vehicle Battery Packs 14.4V
Parallax Board of Education Rev C.
this wire needs to be added to the BOE
DB-9 (F)
RS232
Basic Stamp BS2-SX
Sout
Sin
ATN
Vss
P0
P1
P2
P3
P4
P5
P6
P7
Vin
Vss
Rst
Vcc
P15
P14
P13
P12
P11
P10
P9
P8
Vin
5V Voltage Regulator
Vcc
Memsic 2125 dual axis thermal accelerometer
To
Yo
Gnd
Vcc
Xo
Gnd
900Mhz Video Transmitter
Vcc
Gnd
Video In
900Mhz Video Transmitter
Vcc
Gnd
Video In
Vcc
Gnd
Video Out
PC-171XS CCD Color Camera
Vcc
Gnd
Video Out
PC-171XS CCD Color Camera
Gnd
Trigg
Eout
Vcc 5v
Devantech SRF04 Range Finder (chip-side view)
Gnd
Trigg
Eout
Vcc 5v
Devantech SRF04 Range Finder (chip-side view)

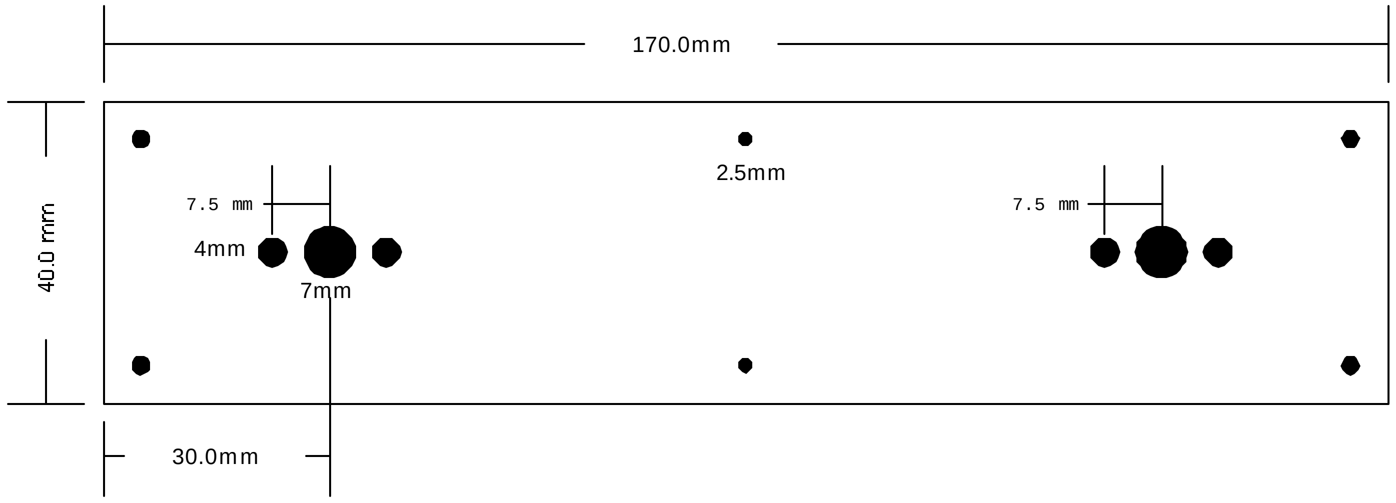
D.A.V.E. PC Interface



Camera mount for DARPA vehicle



1/8 inchthickstyrenesheet



4x 1/4 inch 2-56 fillister head screws with hex nuts
2x 1/4 inch 4-40 flat head machine screws and nuts
6x 3/4 inch 4-40 flat head machine screws and nuts