Sensor Fusion For Autonomous Outdoor Navigation Using Neural Networks

Ian Lane Davis Anthony Stentz

The Robotics Institute
Carnegie Mellon University
Pittsburgh PA 15213

Abstract

For many navigation tasks, a single sensing modality is sufficiently rich to accomplish the desired motion control goals; for practical autonomous outdoor navigation, a single sensing modality is a crippling limitation on what tasks can be undertaken. Using a neural network paradigm particularly well suited to sensor fusion we have successfully performed simulated and real-world navigation tasks that required the use of multiple sensing modalities.

1 Introduction

Our group at Carnegie Mellon University researches autonomous navigation for wheeled vehicles using the HMMWV (High Mobility Multi-Wheeled Vehicle - pronounced "humvee"), a four-wheel-drive military ambulance. Some of the most successful work to come out of the group has included road following using CCD (charge coupled device) [10] cameras and cross country (XC) navigation using a scanning laser range finder [7]. The long term goal of each of these projects has been to engineer a vehicle capable of performing fully autonomous tasks in a given environment.

But current systems have significant limitations. A single CCD camera is sufficient for road following, but not for extended road driving tasks, and a laser range finder is sufficient for obstacle avoidance, but not for cross country navigation in "interesting" terrains.

1.1 Fully self-contained autonomous navigation

In both the road following and the cross country navigation tasks, as well as in many other tasks (such as space exploration), *fully self-contained* navigation is either very desirable or entirely necessary. Although we can enhance capabilities with "top-down" data such as maps, the techniques which have worked so well to date for road-following and obstacle avoidance have been "bottom-up" techniques, which take the rawest of locally available data and turn this into the necessary motion control commands. Inherent in this concept is rich sensing. Thus, our goal is to integrate multiple sensors in order to achieve capabilities not previously attainable in autonomous navigation systems.

1.2 Prior work & our approach

The approach taken in our research is informed by much of the work in cross country navigation [1][7], work in neural networks for navigation [2][8][9][10][13], and work in modular neural networks [3][4][5][6][12].

Our current experiments with sensor fusion use backpropagation neural networks [11]. The basic elements of a neural network are nodes and connections. Nodes have activation levels, and connections have weights. The nodes are frequently organized into three groups: inputs, outputs, and hidden units. The network learns by having a pattern put into the input nodes and adjusting the weights of all of the connections until the output nodes show the desired output pattern.

Why do we choose to use neural networks? For an excellent discussion of the suitability of neural networks for the general navigation task see [10]. The primary issue is that we do not necessarily know all of the features in sensor space which affect our navigation decisions. Neural networks, when properly trained, can automatically select and weight the most important features of an environment, with the added bonus of being able to tackle a radically different environment with just a change of training data.

The architectures of the networks we examine in this work are of two flavors: monolithic networks and modular networks. In the typical monolithic neural network approach, we give the network all of the sensor data as inputs and allow the network to develop internal representations (by adjusting weights between nodes) which allow it to perform adequately within the context of the experiences we expose it to.

However, we do not always know how to interpret the internal representations that the neural network generates. The network may even cue on features that are artifacts of

0-8186-7108-4/95 $4.00

our training data and not central to the task. What if we know ahead of time that some feature in the sensor space is useful for navigation? For example, we know that the edges of the road are good clues to use in on-road navigation. A typical neural paradigm might result in a network learning to find the edges of the road. Or it might not. Exactly how much of our resources (time, memory, computing power, patience) are we devoting to learning what we already know?

These issues have led us to examine modular neural networks as an alternative. The modular architecture we use addresses the integration of a priori knowledge and may prove to be more flexible than a monolithic approach as the complexity of the tasks increase. Our modular system is the Modular Architecture Multi-Modal Theory network (MAMMOTH). MAMMOTH is both a network architecture, and a training paradigm.

A MAMMOTH system consists of two segments, the feature level and the task level. The feature level is a set of Feature Neural Networks (FeNNs) trained to recognize specific features in whatever sensor modality serves as their input source. Thus, there might be one or more networks in the feature level devoted to known-to-be-useful features in the color video modality. Other FeNNs might recognize features in laser range finder sensor space or in the input space of any modality.

On top of this feature level is the task network which unites the feature networks, the Task Network (Task Net). The task level uses the information from the feature level networks' hidden layers as inputs to a network trained to perform the navigation task. For a sketch of a MAMMOTH network see Figure 2.

2 The task at hand

Our task is to achieve autonomous navigation with the HMMWV. We want the vehicle to wander safely around the terrain avoiding obstacles (but not heading to particular goals). We want our system to choose an appropriate steering direction (throttle is not computer-actuated). The HMMWV has all of its computing, power, and sensing on-board. We use both primary sensors: a color video camera and a scanning laser range sensor.

Most work was tested in one of two environments: on a square kilometer of a "virtual" 2.5D world or at 2 kilometer by 2 kilometer outdoor testing site on top of an old slag heap. Alsim, our simulator [7], consists of a map with elevation and color (RGB) in each cell. The initial elevations are generated with a triangle based fractal, and coloration is based on a similar fractal. This creates a world with varying elevation and patches of "vegetation" (a.k.a. green stuff). We can add onto this world a variety of colored geometric shapes and dirt or paved roads.

The real world slag heap has sections that range from very sparsely vegetated to intraversible due to trees and thick bushes. There are natural and man-made ridges and mounds of earth, cliffs, large semi-permanent puddles, and remnants of the foundations of a few buildings. A dirt access road runs through a large section of the slag heap (looping around to almost 4k total length). A rough estimate would be that less than half of the total surface area of the slag heap is traversible with the HMMWV. Most current systems are even more limited in the sections they can reach because they do not use multiple sensors and correspondingly cannot deal well with vegetation.

2.1 System issues

In both Alsim and real world experimentation (and especially in the real world), speed of processing is important, and in order to keep the whole control loop going at slightly faster than a convenient (if slightly arbitrary) goal of 1 Hz we reduced the images from their original high resolution to lower resolutions. This cut down on processing time for the neural network simulations. Fortunately, in order to achieve the goal of proof of concept for the sensor fusion mission, we did not need to detect *every* obstacle which could be harmful to the HMMWV. We counted on the human safety driver to keep the vehicle from harm due to obstacles below sensor resolution. We felt safe in using laser images which were 16 x 64 pixels, and camera images which were 15 x 20 in each of the red, green, and blue bands (reduced from 64 x 256 and 480 x 640 respectively). This meant that most of our runs were stopped at some point when the vehicle encountered an obstacle it physically could not see.

3 Network architecture

The neural networks, too, had to be tailored for execution in real-time. Generally, this was not a problem because the biggest implication for the networks was that the number of hidden units (see [11] for a full explanation), had to be kept to as few as possible to accomplish the task. But neural networks are function approximators, creating a mapping from the input to the output. The number of hidden units roughly corresponds to the dimensionality of the function you try to fit to the data. If you try to fit a high dimensional function to a data set which could be represented better by a low dimensional function, you will fail to extrapolate from or (worse) interpolate between the provided data points. Thus, you want to use as few hidden units as necessary anyway.

3.1 Monolithic architectures used

We used two monolithic neural network architectures which performed almost identically. Most of the time we used a simple three layer feed-forward network with 1860 inputs from the laser and the CCD camera, five hidden units, and eleven output units. The outputs and hidden units used were based on results from Pomerleau [10] and from experimentation. The output activations were done in a gaussian form, such as Pomerleau used, so that the activation was highest on node corresponding to the desired steering direction and sloped off to either side. An additional hidden unit of five nodes was sometimes added between the first hidden layer and the outputs in order to make the complexity of connections similar to those found in the MAMMOTH network used. See Figure 1. for a diagram of this network.

To train the monolithic network we gathered a training set of 300 exemplars (an exemplar consists of an input image to output direction pair). Then the network was trained in two to five hundred epochs (an epoch is one pass through the whole training set in batch training [11]) until it performed adequately for our tasks.

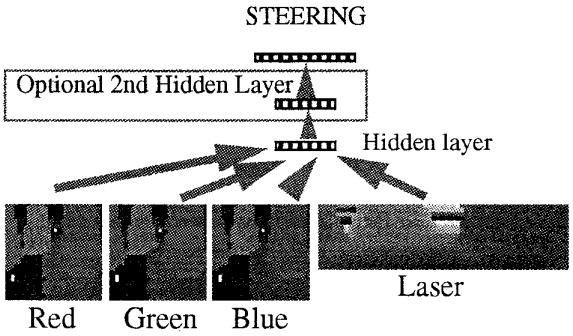

Figure 1: Monolithic navigation network

3.2 The specific MAMMOTH architecture

For all of our modular network experiments we used the same MAMMOTH architecture, which had the same number of inputs and outputs as the monolithic network, and which had roughly the same number of connections. In this case, all of the CCD inputs (in red, green, and blue) are connected to one set of hidden units which are connected to one set of outputs. This feature network, or FeNN, was trained to find features relevant to driving in the color image space. Another FeNN connected the range image inputs to a second set of hiddens and to another output. The task network with a set of hidden units and its own output layer sat above this and was connected to the hidden units of the two FeNNs. See Figure 2

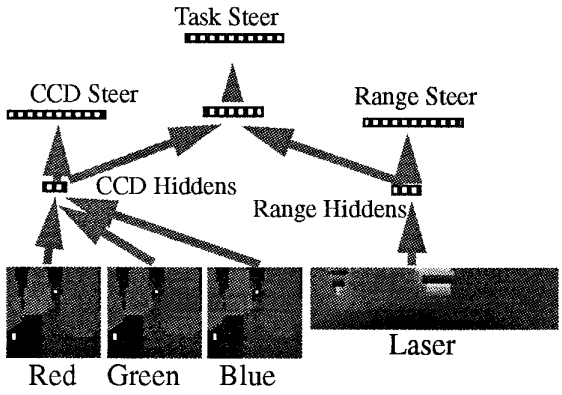

Figure 2: MAMMOTH network for navigation

The interesting aspect of the FeNNs is that they are trained before the task network is trained, and then their weights are frozen. The task network is then trained with the hidden units of the FeNNs as its inputs. If the FeNNs are well designed and the training sets for the FeNNs are constructed carefully, you can force these hidden units to represent a narrow portion of the input space. For our experiments the task network was trained with the exact same training sets as the monolithic networks.

4 Experiments and results

The two experiments in Alsim took the road following work of Pomerleau [10] a step further by adding obstacle avoidance with the laser range sensor. Likewise, our two real world tests augmented obstacle avoidance with CCD camera data.

4.1 Alsim: Stationary road-colored obstacles

The first experiment involved following a road while avoiding obstacles. The obstacles were stationary and black, which was the same color as the road. The vehicle had to use both sensors (color video and laser range) to accomplish this goal. Figure 3 shows a map of the simulated terrain. There are block shaped obstacles are at points A, B, & C. A black pit is at point D. The road we want the vehicle to follow is the paved road with the line down the middle. There is a "dirt" road (brown) running east-west, and various geometric obstacles off-road.

Both monolithic and MAMMOTH networks were used for this test. We collected training exemplars once per second. For the monolithic network's training set and the Task Net of the MAMMOTH network, the vehicle was driven (using the mouse to steer) over the road several

times in each direction. Additional exemplars were gathered by driving off of the road (with data collection off) and then driving back onto the road; these exemplars were necessary so that the simulated vehicle could recover from small navigational errors. All the while, the vehicle was steered so as not to hit any of the three obstacles on the road and to drive on the road when possible.

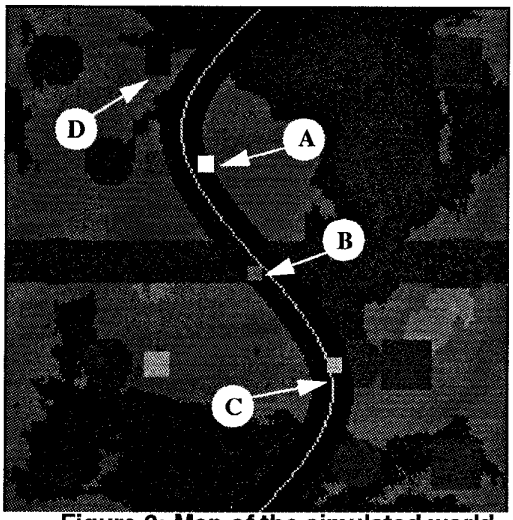

Figure 3: Map of the simulated world

For the monolithic network we were able to train just with this data. However, for the MAMMOTH network, we first trained the FeNNs. One FeNN was for road-navigational features using the simulated CCD camera and the other FeNN was for obstacle-avoidance-navigational features using the simulated laser range finder.

Alsim gave us the power to easily make training sets for FeNNs. The nature of a FeNN is that we want it to learn internal representations (weight patterns) that represent a narrow and known set of features. In Alsim we can force a FeNN to learn only what we want by creating worlds whose only features are those we wish to learn. To train a FeNN to recognize low level video features of roads appropriate for navigation and not to key off of extraneous image features, we make a world in which the only feature is a road. Likewise, to train an obstacle avoidance FeNN, we make a world in which the only features are large obstacles to be avoided.

For the MAMMOTH network, 300 training exemplars were gathered in both the road-world and the obstacle-world. The respective FeNNs were trained with these and then the weights to the outputs of these FeNNs were frozen. Finally, the same training set used on the monolithic network was used to train the Task Net part of the MAMMOTH architecture.

Performance was excellent on this task for both networks. For both training and testing the speed of the simulated HMMWV was held perfectly constant, and the vehicle was able to stay on the road and avoid obstacles for over an hour and a half, which corresponds to about an hour at a slow speed (3mph) in the real world. Another option of Alsim allowed us to "warp" the vehicle to a random location and orientation in the simulated world, and three times out of four the vehicle resumed traveling on the road when it intersected the road again. About every fourth time the vehicle would intersect the road at an almost right angle and that was sufficiently different from what was in the training set that it would respond improperly and drive straight across the road.

4.2 Alsim: brightly colored obstacles

Another simulator task highlights one of the pitfalls that hinders researchers using neural networks, and the results suggest two solutions. The goal was again to stay on the road and to avoid the static obstacles, but this time the three obstacles were colored blue, green, and yellow from north to south along the road. Training sets were gathered in the same manner as in the earlier task.

The results were similar to those of the previous section. The particular training set for this task produced slightly sloppier driving habits in the simulated HMMWV, and this was evidence by the occasional (maybe 1 in 8 occurrence) of driving fairly widely around the blue box at point A and then heading straight into the pit at point D. This behavior was bad. However, the explanation and fix are simple. This happened because the vehicle had not driven so far to the left on the road near point A before and thus had not seen the pit very well before. The pit and the walls of the pit were black - the same color as the road. If the vehicle had just avoided the block at point A widely, it would see the black of the pit better than the black of the road and head that way. The fix is simply to add examples of avoiding pits to the training set.

A more interesting result came up when the networks trained with the blue, green, and yellow blocks were tested on a simulated world which had three bright red blocks at the same locations. The vehicle went off the road and into the pit much more frequently. With the monolithic network there was at least a two in three chance of going off of the road, and with the modular network there was approximately a one in three chance. The implication is clear. The person training the network had correctly driven more closely around the blue box at point A than the yellow box at point C in the training. And both networks had learned to go left around blue boxes more tightly than around yellow boxes. With the red blocks, the network drove more loosely around the block at point A, and saw the pit that much more often. Another symptom was that

the vehicle frequently went off of the road at point B. There was an almost 100% occurrence with the monolithic network and at least a 50% chance with the modular network.

Both of these problems have the same explanation: the yellow block in the first colored-block world was the only block with a red component in RGB. The network learned to go wide around the yellow block since it had to go around it on the left to stay on the road and there was also a left turn immediately after the block on the road. This made the network go widely to the left of the block at point A and often to the left of the block at point B.

A simple fix is training sets which have examples of blocks of every color on both sides of the road. But note that although the training set for the MAMMOTH task network was identical to that for the monolithic network, it helped that the lower level representations were strictly separated into obstacles in laser range space and representations for road following. By designing our FeNNs to represent narrowly what we want, we decreased the amount that we keyed on incidental features.

4.3 Real world: additive obstacle test

If everything in the world was rigid, we could avoid everything[7]. However, sometimes we must drive over the vegetation when we have no other decent route. The laser range finder alone cannot find vegetation, so we must use an additional sensor, the color video camera.

The first task has a simple goal: don't hit big solid obstacles and also don't hit vegetation. We call the first task the additive obstacle task because the additional sensor only adds new things to avoid. We trained and tested in an area of the slag heap containing an access road which often had vegetation and/or earth mounds on one or both sides.

We used the same network architectures as for the simulated tasks. It is clear that generating appropriate training sets is significantly more difficult. For the Task Net part of the MAMMOTH network and for the monolithic network, we drove the HMMWV around the terrain. This would have been simple, except we needed examples of recovering from mistakes, which meant we had to make mistakes (with data collection off), and mistakes are that much trickier in a five ton truck than in a simulated five ton truck. Consequently, speed varied, and the training set was less rich in recovery examples.

But training the FeNNs was the real challenge. In the real world we cannot simply train in a world that contains the narrow set of features we wish to detect. The laser range FeNN was trained to avoid obstacles, as it had been in Alsim. There were several large mounds of earth sufficiently far from vegetation which we used as our prototypical obstacles, and we trained by avoiding these mounds as we had done in Alsim.

We also trained a FeNN to drive around avoiding vegetation. In the real world, though, images containing vegetation often contain other things; additionally, vegetation in the real world is not nearly as homogeneous as any feature we might want to recognize in the simulated world. We had to develop a training set that contained a rich set of examples of the HMMWV steering away from all of the types of vegetation we might encounter.

Given these difficulties, the FeNNs we trained both worked quite well, and the monolithic network and the Task Net performed even better. In fact, our two main problems were that there were obstacles that simply didn't show up at the resolutions we used in sensor space (such as sharp steel bars and jagged cinder blocks) and that there were culs-de-sac at the slag heap.

Both of the FeNNs would allow the HMMWV to drive for runs of up to 0.6 miles before we ended up in a cul-de-sac or had something dangerous right in front of us that could not be seen at low resolution. Another failure mode was the result of uneven ground or territory not covered in the training sets for the FeNNs. The most frequent failure mode for the CCD FeNN was driving right up to a cliff that was the same color as the road. Likewise a common failure mode for the laser range FeNN was driving into low vegetation that got progressively more dense until we gradually appeared in an area where the vegetation was dense enough that no correct steering direction could be determined from range images.

Combining the two sensor modalities had the desired results. Typical runs of the HMMWV were around 1 mile or more before we encountered a failure mode. Both the monolithic network and the MAMMOTH network performed very well, though there is not yet enough data for a significant quantitative comparison.

4.4 Real world: conflicting obstacle test

The final test was one which we've just begun to tackle on the real vehicle. The location of the test was a section of roughly 1600 square meters on the slag heap which has numerous man-made mounds of earth spaced just a bit wider than is necessary for the HMMWV to drive through. The mounds of earth are around three feet high and five feet in diameter, and between them vegetation grows out of control. Our task was to drive the HMMWV through this obstacle course. The laser range sensor generated images in which the vegetation looked like the mounds of earth, so the video camera was needed to differentiate between safe and unsafe steering directions. However, in order to function outside of the densely vegetated areas, we needed the laser range camera for avoiding isolated mounds.

Once again our testing used both the monolithic neural

network and the MAMMOTH network. The CCD FeNN from the MAMMOTH network was trained with freshly generated images of the new test region, the laser range FeNN was trained with the same exact training set used for the additive obstacle test, and the Task Net and monolithic network were trained with a set that was a concatenation of the training set for the CCD FeNN and an equal number of exemplars from areas of less dense vegetation.

In the few outings we've had, the monolithic neural network was largely unsuccessful. The modular MAMMOTH network was able to go through the test region about 40% of the time. One cause of failure was training and testing on different days with different lighting. The other major cause was at the very slow speeds we had to use, the vehicle's front-mounted sensors would often clear an obstacle before the wheels did (and the current reactive-only system would think all was clear).

5 Conclusions

With this work we have achieved new capabilities in cross country navigation and sensor fusion, and have continued an exploration of an alternate neural network paradigm which produces more understandable and reusable neural networks.

For cross country navigation, we have added the ability to handle vegetation intelligently. We have succeeded in training a network to avoid vegetation and the work in simulation and the preliminary field work suggest that we can also train the neural network to choose to drive over vegetation when the other options are worse.

The implications of this work for sensor fusion in general is that we can feel confident to add new sensor modalities to a given task. A monolithic neural network is capable of learning to fuse the sensing modalities, but the MAMMOTH architecture gives us even more control over how to utilize the data provided in any given modality. Using appropriate FeNNs we can add some of our a priori knowledge of useful features and at the same time increase our ability to interpret how each sensing modality contributes to the task performance. Also, by tightly focussing the lower level internal representations of the network through the use of the FeNNs we are able to avoid cross-modality interference that can result in poor performance from keying on incidental features (such as the redness of a block in the simulated world).

In the near future, we will continue work with all of our tasks, and plan to add goal-seeking behavior as soon as possible. For all of our tasks, we will optimize our computing use to allow increased sensor resolution. Furthermore, we are currently exploring optimization techniques for the neural network architectures and topological analyses of the mappings from sensor space to control space.

6 Acknowledgments

This work has been supported in part by a National Science Foundation Graduate Research Fellowship. The equipment used is provided in part by DACA76-89-C0014, Topographic Engineering Center, Perception for Outdoor Navigation and DAAE07-90-C-R059, TACOM, CMU Autonomous Ground Vehicle Extension.

7 References

[1] M. Daily, et al., "Autonomous Cross Country Navigation with the ALV," *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*: 718-726, 1988.

[2] I. L. Davis, "Neural Networks For Real-Time Terrain Typing," Carnegie Mellon Tech Report CMU-RI-TR-95-06, 1995.

[3] M. Gluck and C. Myers, "Hippocampal Mediation of Stimulus Representation: a Computational Theory," To appear in *Hippocampus* (in press 1993), 1993.

[4] J. B. Hampshire and A. H. Waibel, "The Meta-Pi Network: Building Distributed Knowledge Representations for Robust Multi-Source Pattern Recognition," IEEE-PAMI, 1989.

[5] R. A. Jacobs, M. L. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation*, 3:79-87, 1991.

[6] T. Jochem, D. Pomerleau, C. Thorpe, "MANIAC: A Next Generation Neurally Based Autonomous Road Follower," *Proc of International Conference on Intelligent Autonomous Systems* (IAS-3), Feb 15-18, 1993.

[7] A. Kelly, "A Partial Analysis of the High Speed Cross Country Navigation Problem," Carnegie-Mellon University Ph. D. Thesis Proposal, 1993.

[8] M. Marra, R. T. Dunlay, and D. Mathis, "Terrain Classification Using Texture for the ALV," Proceedings of the SPIE Conference on Mobile Robots, 1992.

[9] L. Meeden, G. McGraw, and D. Blank, "Emergent Control and Planning in an Autonomous Vehicle," *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, 1993.

[10]D. Pomerleau, *Neural Network Perception for Mobile Robot Guidance*, Ph.D. Dissertation, Carnegie-Mellon University Technical Report CMU-CS-92-115, 1992.

[11]D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Ed. MIT Press, 1986.

[12]F. J. Smieja and H. Mühlenbein, "Reflective Modular Neural Network Systems," Technical Report, German National Research Centre for Computer Science, March, 1992.

[13]W. A. Wright, "Contextual Road Finding With A Neural Network," Technical Report, Sowerby Research Centre, Advanced Information Processing Department, British Aerospace, 1989.