

Genetic Programming for Real World Robot Vision

Martin C. Martin

Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA, mcm@mit.edu

Abstract

The vision subsystem of an autonomous mobile robot was created using a form of evolutionary computation known as genetic programming. In this form, individuals are algorithms represented as parse trees. The primitives of the representation were specifically chosen to capture the spirit of existing vision algorithms. Thus, the evolutionary computation can be viewed as searching roughly the same space that researchers search when developing their system using trial and error.

Traditional image operators such as Sobel magnitude and a median filter were combined in arbitrary ways, and images from an unmodified office environment were used as training data. A hand written obstacle avoidance algorithm used the output of the best vision algorithm to avoid obstacles in real time. It performed as well as existing, hand written combined navigation and vision systems.

1. Introduction

Somewhat surprisingly, while there have been many successes in mobile robot obstacle avoidance using sonar, obstacle avoidance using *vision* has received much less attention. Yet vision can potentially give much greater information about the world, if we can just get the information out of images. Rather than write such algorithms by hand, this work uses genetic programming to create algorithms that work empirically in a given environment. Significantly, the representation was intentionally chosen to capture the spirit of existing, hand written algorithms.

An earlier iteration of this framework was presented in [10]. The current work improves on that in a host of ways. Among other things, the earlier work required a hand written, poorly performing “seed” individual be inserted in the initial population. The refinements reported here, among other things, obviate the need for seed.

2. Previous Work

Many have explored the use of evolutionary computation for creating low level visual routines such as edge detectors and interest operators that return a result for

every pixel. In particular, Ebner [3] used Genetic Programming to evolve interest operators such as the Moravec interest operator. In contrast, Johnson *et al.* [5] evolved an entire visual routine to find the hands in binary images of people. Their evolved programs correctly classified up to 93% of images, better than the best algorithms they were able to write by hand.

Teller *et al.* [17] evolved programs to classify images which contained functions to return the min, max, mean and variance over a rectangle. They also made use of an array of memory, as an alternate way of maintaining state. Cliff *et al.* [1] evolved programs that allowed the Sussex gantry robot to move toward a white square and not a white triangle in an otherwise black world.

Somewhat surprisingly, there have only been a handful of complete systems that attempt obstacle avoidance using only vision in environments that weren't created for the robot. Larry Matthies' group has built a number of complete systems, all using stereo vision [12]. Their algorithm has been tested on both a prototype Mars rover and a HMMWV. The Mars rover accomplished a 100m autonomous run in sandy terrain interspersed with bushes and mounds of dirt [11]. The HMMWV has also accomplished runs of over 2 km without need for intervention, in natural off-road areas [13].

Ratler [7] used a stereo vision algorithm to do autonomous navigation in planetary analog terrain. Travelling at 50 cm/sec over 6.7km the system had 16 failures, for a mean distance between failures of 417m. David Coombs' group at NIST has succeeded using optical flow in runs of up to 20 minutes without collision in an office environment [2].

Ian Horswill's algorithm [4] assumes that the floor is textureless and that obstacles aren't, and finds the lowest “non-floor” area in each column. His system ran for hundreds of hours without failure. Liana Lorigo's algorithm [8] assumes the bottom of the image represents clear ground, and searches up the image for the first window that has a different histogram than the bottom, turning left or right depending on which side has the most floor. Ulrich and Nourbakhsh [18] took the floor to be the part of a previous image that had since been traversed.

Illah Nourbakhsh has used depth from focus for robot obstacle avoidance [15]. Three cameras, focused at dif-



Figure 1: The vision subsystem semantics. Given a column in the image, the output of the evolved program was interpreted as the boundary between ground (lower white line) and the lowest non-ground (upper white line). Show is the ground truth in six different columns.

ferent distances (near, middle and far), image the same scene. Whichever image is sharpest is the most in focus, so the objects are roughly at that distance. In hundreds of hours of tests, the robot has avoided stair cases as well as students, often running down its batteries before a collision.

3. Experimental Setup

All experiments used the Uranus mobile robot of the Mobile Robot Lab, Carnegie Mellon University. While the robot had a three degree of freedom base, motions were generally restricted to moving forward while turning, i.e., to those possible with Ackerman steering. Its sensors consisted of a b/w analog video camera and a ring of 24 sonar sensors. Both online and offline processing were done by an off board dual 700MHz Pentium III computer running BeOS.

The controlling software consisted of two subsystems, the *vision subsystem* and the *obstacle avoidance subsystem*. The semantics of the interface between them were completely specified. The vision subsystem took in an image, that is, a 320×240 array of greyscale values, and the horizontal location of a column. It returned an estimate of the lowest non-ground pixel in that column, see Figure 1. Assuming objects touch the ground and that the ground is roughly flat, the height of the lowest non-ground pixel is a monotonic function of the distance from the robot. The obstacle avoidance subsystem took a number of such estimates from the current image and returned a direction to travel. No state was maintained from one image to the next or between columns within an image, which means all programs were reactive.

Although these semantics were externally imposed, they weren't arbitrary. A dense depth map was intentionally avoided, inspired by arguments that representation should be used sparingly. Instead, the semantics were kept close to the semantics of the most popular sen-

Record Video	Robot, Real Time
Learn Offline	Genetic Programming
Build Obstacle Avoidance	By Hand
Validate Online	Robot, Real Time

Figure 2: The control software was constructed in four phases. First, a set of images were collected from the target environments. Second, genetic programming was used offline to create a vision subsystem. Next, an obstacle avoidance subsystem was written by hand. Finally, the combined system controlled the robot.

sors for mobile robot obstacle avoidance, sonar and laser range finders, which return distances.

Figure 2 outlines the process. The vision subsystem was created first using a standard supervised learning framework. The obstacle avoidance subsystem was then written by hand to be robust to the types of errors made by the vision subsystem. Knowledge about the frequency and nature of failure modes could guide the design of the next version of the vision subsystem.

The vision subsystem was constructed using Genetic Programming [6]. Genetic programming is a genetic algorithm in which individuals are programs, represented as parse trees. The set of node types is chosen by the programmer and is dependent on the problem domain. Random programs are created by choosing a random node type for the root, then choosing a random node type for each of its arguments, continuing recursively. A fixed number (say 1000) are created this way and each one is assigned a scalar *fitness*. Individuals are then randomly chosen, with the fitter individuals more likely to be chosen, and are either copied verbatim into the next population (*replication*) or a random subtree is chosen in two different individuals and swapped (*cross-over*). Once 1000 individuals are created this way, the original individuals are discarded, the new individuals are assigned fitness and the process repeats.

The camera was calibrated using the system described in [14], then mounted on the robot and pitched 51 degrees down from horizontal. The training set was gathered while the robot navigated using sonar, not vision. The view from the camera, a series of 320×240 greyscale images, was passively recorded. A detailed description of the algorithm used to navigate from sonar can be found in [9].

Three datasets were used. The first two consisted of images from two hallways in different buildings, Newell Simon Hall (NSH) and the Field & Mobile Robotics Building (FMRB). The third data set consisted of every other image from the first two. Both environments

Table 2: Evolution Parameters

Objective	Given an image and a horizontal position within it, return the first non-ground pixel in that column.
Architecture of individuals	Two trees, each with a single <i>iterate</i> node at the root.
Fitness cases	Six columns in each of 65-71 images. 390-426 total.
Raw fitness	The percentage of fitness cases “correct.” A fitness case was “correct” if the pixel location estimated by the evolved program was within 10 pixels of the author’s determination of the ideal answer.
Standardized fitness	100% minus raw fitness.
Parameters	51 generations, population size of 10,000, tournament selection with tournament size of 7, ramped-half-and-half with min size 6 and max size 9.

ters were inspired by Teller *et al.* [17]. An example *iterate* tree is shown in Figure 3.

Each tree had a single *iterate* node, always at the root, and this rule was enforced during creation and crossover of individuals. In a simplification of earlier work [10] each individual consisted of two *iterate* trees, and the second tree could use the results of the first tree. *Iterate* nodes do not return a value in the traditional sense. Instead, the final values of the five registers are used as five return values from the tree. The result of the entire individual was simply the final value of the *a* register of the second tree.

The evolution run started by rectifying the images to an ideal perspective projection and cropping them to a horizontal field of view of 83° using the system in [14]. The image operators were then precomputed at every location of every image.

The other parameters used during evolution are summarized in Table 2. On a dual 700MHz Pentium III the times to assign fitness to a single individual varied widely but averaged 725 msec. The time for an entire run of 51 generations also varied widely, averaging approximately 20 hours.

4. Training Results

All three experiments produced individuals which achieved a fitness of greater than 85%. They did this despite burned out lights and other effects that caused the carpet’s average intensity to vary from zero to at least



Figure 4: Performance on the training data of the best individual from the FMRB hallway experiment.

145 out of 255; despite large gradients caused by imaging artifacts; despite moiré patterns of image noise; and despite the shadow of the robot.

On the FMRB training data, the best constant approximation, ignoring both the image and the desired column, was to return $\text{image-max-y} - 10$, i.e. 229, to get 101 answers correct for a fitness of 23.7%. In all but two of the runs, the best individual in the initial population did worse than that, returning $\text{image-max-y} - 3$ and achieving 21.6%. The other two runs achieved 29.3% and 36.4% in the initial population. The runs on the other two datasets were similar, occasionally but rarely performing better than the best constant approximation.

The best individual from all FMRB runs achieved 91.78%. Some examples from the training set are shown in Figure 4. While hard to see, the white squares are the size of the window chosen by evolution. In the two upper images, all six fitness cases were determined correctly. In the lower left, one of the two on the filing cabinets was higher than desired, although a difference this small would not affect obstacle avoidance significantly. In the lower right, the rightmost column should be at the bottom of the image instead of near the top.

The other two conditions did almost as well, with the best individual from all NSH runs achieving 90.51%, and the best from the combined data set 84.8%. In all three experiments the errors were transient with the exception of the stripe of red carpet in Newell Simon Hall, which was uniformly considered an obstacle when near the camera.

The best individual from the FMRB condition was simplified, partially by hand and partially automatically. The resulting program was quite readable (Figure 5.) The two branches used two very different techniques. The result producing branch was essentially

Result Producing Branch:

Iterate-Down, 3x3 window, centered on desired column:

```
if y ≤ 9 then
  a := y;

if second-branch-b > 9 then
  a := y;

if median > 35.4444 then
{
  if gradient > 413.96 then
    a := y;

  if gradient > 239 and (gradient/239)4 > diag-grad then
    a := y;
}
```

Second Branch:

b (initial value) := 3.40282e+38

Iterate-Up, 3x3 window, centered on desired column:

$$b = \frac{b(1+h) - (1+h+h^2+h^4)\text{desired-x}}{h^5}$$

Where h is horizontal gradient.

Figure 5: A simplified version of the best individual from the FMRB experiment.

a decision tree, although one of the decisions has a non-linear boundary in the space spanned by two image operators. The other branch, which detected obstacles at the bottom of the image, was a recurrent mathematical function involving both a single image operator and the location of the column.

This division into two cases—gradient based boundaries and objects that touch the floor—was discovered automatically. Nothing in the representation suggested the two different cases, nor that the two branches should each tackle a separate case. Instead, this reflects a natural dichotomy in the images: at the bottom of the image a program must detect the presence or absence of an object, but in the middle of an image it must detect the *transition* between ground and non-ground. During a transition there is likely to be a significant gradient, whereas at the bottom of the image there isn't. Similarly, ignoring the gradient when the image intensity was low, which ignored artificial gradients caused by a quirk in the imaging process, was not suggested by the represen-

tation either. These are examples of the genetic algorithm simultaneously exploiting regularities in both the problem domain and the representation.

The other best-of-experiment individuals were similarly simplified to discover how they worked. Evolution had exploited a number of techniques, including a sequence of if-then conditions similar to a decision tree but involving non-linear combinations of up to five different image terminals. In all cases, the bottom of the image was handled using different code than the rest of the image. The mechanisms for this varied; in the FMRB experiment it used two different branches for the two conditions, whereas in the NSH and combined experiments a multitude of *if-le* statements were used to run different code at different locations. Interestingly, the raw image was never used, although the median filtered image was. All directional gradients of the image intensity were used except the vertical.

5. Validation On The Robot

All three individuals generalized surprisingly well when run on new video from the same camera in the same hallway. They were relatively insensitive to the pitch and the horizontal location of the column in the image. With the camera set to automatic gain they also provided acceptable results over a wide range of iris settings. They detected objects that weren't present during training, such as chairs or people, with only a little less fidelity than they detected walls. They were also fast, running at about 10 Hz on a 700 MHz Pentium III.

After the offline learning, a hand written obstacle avoidance algorithm used the estimates to decide speed and direction to travel. This algorithm was very similar to the one used during data collection, except that its inputs came from vision, not sonar. It determined speed from proximity to the nearest object, and determined direction of travel by fitting lines to points on the left and right sides of the robot. The three different vision subsystems, one from each experiment, all used the same navigation subsystem. The best evolved algorithm was run on twelve columns in the image, twice as many as used in training, as six columns allowed rather large objects to pass between the columns. Only six and twelve columns were tried, and twelve found to be sufficient. A detailed description of the algorithm can be found in [9].

This algorithm was created by hand using traditional iterative design, and is still far from optimal. It is a natural application for simulated evolution, which is likely to do significantly better.

The robot was then run in the same environment(s) it was trained in. Videos of this *online validation* can be found at www.metahuman.org/martin/Dissertation. These routes retraced the path of the training set and then

continued much further. They included views of the same hallway from the opposite direction, as well as similar areas never seen during training. Objects were present that were not present during training, such as chairs, trash cans and people.

The evolved algorithms worked well despite months of wear & tear on the carpet. Most errors were transient, lasting 1 or 2 frames, even when the camera was stationary. Most errors were conservative, i.e., declaring that a pixel at the bottom of the image was non-ground when it was, in fact, ground. It worked on a range of iris settings and camera tilts and didn't overfit to column location.

The area of awareness with vision was approximately 83°, and entirely in front of the robot's base. In the reactive framework described here this makes it almost impossible to successfully navigate doorways, especially since the robot is only a few inches narrower than them. However, it performed very well at corridor following and avoiding obstacles such as people and chairs.

There were a few persistent errors. It was sensitive to small strips of paper or shiny pieces of metal. The red carpet, which was handled poorly even on the training data, was classified as an obstacle when nearby, causing the navigation to consistently treat it as a wall. Other errors would fool the obstacle avoidance system only occasionally, although these were responsible for most collisions.

In 5 validation runs, the mean distance and time to collision for the FMRB individual was 133 meters, 20:01 minutes, and the longest was 260 meters, 39:32 minutes. These are comparable to the published performance of the hand-written systems described in the Previous Work section.

Not surprisingly, the vision subsystem from the combined data set performed a little worse in each environment than the vision system developed from data only in that space. It did not have a new class of error, but instead was more likely to make one of the errors made by the other vision subsystems. In the FMRB hallway, its average distance and time to collision were 72 meters and 10 minutes respectively.

6. References

[1] D. Cliff, P. Husbands and I. Harvey. Evolving Visually Guided Robots. In Proc. of SAB92, the Second Intl. Conf. on the Simulation of Adaptive Behaviour. MIT Press, 1993.

[2] D. Coombs, M. Herman, T. Hong and M. Nashman Real-time Obstacle Avoidance using Central Flow Divergence and Peripheral Flow. Fifth International Conference on Computer Vision June 1995, pp. 276-83.

[3] M. Ebner, "On the Evolution of Interest Operators using Genetic Programming," in Late Breaking Papers at EuroGP'98: the First European Workshop on Genetic

Programming, Paris, France, R. Poli *et al.* (ed.), 14-15 April 1998, pp. 6-10.

[4] I. Horswill, "Analysis of Adaptation and Environment." *Artificial Intelligence*, Vol. 73, pp. 1-30.

[5] M.P. Johnson, P. Maes and T. Darrell, "Evolving Visual Routines," in *Artificial Life IV*, Proc. of the Fourth Intl. Workshop on the Synth. and Simul. of Living Systems., R. Brooks and P. Maes (eds.), Bradford Books, 1994

[6] J. Koza, *Genetic Programming*, MIT Press, Cambridge, MA. 1992.

[7] E. Krotkov, M. Hebert & R. Simmons, Stereo perception and dead reckoning for a prototype lunar rover. *Autonomous Robots* 2(4) Dec 1995, pp. 313-331

[8] L.M. Lorigo, R.A. Brooks, and W.E.L. Grimson Visually-guided obstacle avoidance in unstructured environments. *IEEE Conference on Intelligent Robots and Systems* September 1997.

[9] M.C. Martin, *The Simulated Evolution of Robot Perception*, Ph.D. Dissertation and Carnegie Mellon University Technical Report CMU-RI-TR-01-32. 2001, 159 pp.

[10] M.C. Martin Visual Obstacle Avoidance using Genetic Programming: First Results, *Proceedings of the Genetic and Evolutionary Computation Conference*, July 7-11, 2001, pp. 1107-1113

[11] L.H. Matthies, Stereo vision for planetary rovers: stochastic modeling to near real-time implementation. *International Journal of Computer Vision*, 8(1): 71-91, July 1992.

[12] L.H. Matthies, A. Kelly, & T. Litwin Obstacle Detection for Unmanned Ground Vehicles: A Progress Report. 1995.

[13] L.H. Matthies, Personal communication.

[14] H.P. Moravec, DARPA MARS program research progress, <http://www.frc.ri.cmu.edu/~hpm/project.archive/robot.papers/2000/ARPA.MARS.reports.00/Report.0001.html>, January 2000.

[15] I. Nourbakhsh, A Sighted Robot: Can we ever build a robot that really doesn't hit (or fall into) obstacles? *The Robotics Practitioner*, Spring 1996, pp. 11-14.

[16] I. Nourbakhsh, Property Mapping: A simple technique for mobile robot programming. In *Proceedings of AAAI 2000*.

[17] A. Teller and M. Veloso, PADO: A new learning architecture for object recognition. In *Symbolic Visual Learning*, Oxford Press, pp. 77-112. 1997.

[18] I. Ulrich and I. Nourbakhsh, Appearance-Based Obstacle Detection with Monocular Color Vision. In *Proceedings of AAAI 2000*.