
Learning Visual Feature Detectors for Obstacle Avoidance using Genetic Programming

Andrew J. Marek

Department of Computer Science
Washington University
St. Louis, MO 63130, USA
ajm2@cec.wustl.edu

William D. Smart

Department of Computer Science
Washington University
St. Louis, MO 63130, USA
wds@cs.wustl.edu

Martin C. Martin

MIT Artificial Intelligence Lab
200 Technology Square, NE43-933
Cambridge, MA 02139, USA
martin@metahuman.org

Abstract

In this paper, we use Genetic Programming (GP) techniques to learn visual feature detectors for a mobile robot navigation task. We provide results from a number of environments, each with different characteristics, and draw conclusions about the performance and nature of the training and testing data. We explore the utility of seeding the initial population with a previously evolved individual, and compare this to previous work, where a hand-coded individual was successfully used as an initial seed. Our experiments exhibited a peculiar bi-modality in final performance. An individual either performed very well or very poorly, with nothing in between. We analyze this phenomenon, and offer suggestions as to its cause, and how to alleviate the problem. We explore the utility of seeding the initial population with a previously evolved individual, and compare this to previous work, where a hand-coded individual was successfully used as an initial seed.

1 Introduction

Feature extraction from images is a difficult task. Assumptions about the nature of the environments, and how they appear in images often prove to be incorrect, causing the performance of feature extractors to be lowered. In this paper, we use Genetic Programming (GP) techniques [11, 12, 13] to evolve a visual feature detection algorithm to be used for obstacle avoidance on a mobile robot.

Since such algorithms are often fine-tuned to work on their particular training set, we have performed a series of experiments across different environments, each

with its own distinctive characteristics and challenges. By taking individuals evolved to deal with one environment and running them in each of the other environments, we hope to gain a better understanding of what is needed in a training set for optimal performance in this particular task. We also explore the use of a hand-crafted seed individual in the initial population in order to ensure evolutionary success and high fitness levels.

The work reported here is a direct extension of Martin's [16], and attempts to test how well the results previously reported generalize across similar, but different environments. We begin by discussing the particular domain that we are interested in developing feature detectors for, robot obstacle avoidance. We then briefly summarize some related work in the area of in the area of feature detection. Next, we describe our experimental setup, and then give the results of our experiments. Finally, we offer some conclusions from this work and discuss the directions in which we would like to take it.

2 Robot Obstacle Avoidance

Martin [16] used GP methods to learn algorithms that extracted the height of the the lowest non-floor pixel in a given column of an image. In the office environments in which the work was done, this corresponded to the floor/wall interface, and served as a relative distance measure to the wall. The positions of these lowest non-floor pixels were then used as input to a simple mobile robot navigation algorithm, inspired by the work of Horswill [7]. The basic idea of this algorithm is to steer the robot towards the area that is most open. Since the height of the lowest non-floor pixel corresponds roughly to distance, this means steering the robot towards the highest of the features identified. This system resulted in a surprisingly robust navigation behavior. Figure 1 shows examples of the environment



Figure 1: Environment and detected features from Martin [16].

and the identified features. In all of these experiments, fitness was measured by the distance between the actual lowest non-floor pixel and the position in which the program decided it should be.

The previous work used a tree-based representation, common to many GP approaches. The internal nodes of this representation corresponded to standard mathematical and control operations (such as looping). The terminals included five floating point registers, and a pre-defined set of primitive image operations, such as blurring and various edge detectors. During early experiments, it was noticed that many of the successful individuals, much of the run time was spent performing a computation over a rectangular window which iterated over a column or row of the image. To remove the need to learn this control structure, it was explicitly added to the representation in the form of the *iterate* node. This primitive moved a rectangular window over an image, executing particular (learned) code at each location. Its (learned) arguments determined the size of the rectangle, the initial location, the direction and distance to iterate, and finally a piece of code to execute at each location.

Experiments were performed with images taken from several locations in an office environment. In the initial set of experiments, the *iterate* nodes could operate either horizontally or vertically. The best individuals learned in these experiments performed only slightly better than programs that ignored the actual image data and just returned a fixed number. This happened despite a population size of 10,000 individuals running for 51 generations. Individuals were limited to a total size of 6000 nodes.

However, when a poorly performing hand-written program was used to seed the population, much more successful individuals were learned. The best such individual achieved an average error of only 2.4 pixels, and identified 60% of the test cases within 2 pixels of the human-provided ground truth position. Martin reports subjectively that the vast majority of fitness cases were handled more than well enough for obstacle avoidance, and the errors did not follow any particular pattern, suggesting a simple filter would eliminate

most of them.

While many aspects of the seed individual were modified by the evolutionary computation, others were not. In particular, the successful evolved algorithms all iterated vertically, from the bottom of the image to the top or vice versa, just as the seed did. Therefore, in a second set of experiments, horizontal iteration was eliminated and the *iterate* node simplified to take only three arguments: the window size, the horizontal location in which to iterate, and the code to execute at every step. The result producing branch, which had simply returned the result of a single iteration branch, was also eliminated.

These new experiments produced individuals which routinely achieved a fitness of greater than 85%. They did this despite image features such as burned out lights and other lighting effects that caused the carpets average greyscale intensity to vary between values of 0 and 140 (out of 255). They also coped with large gradients caused by imaging artifacts, moiré patterns of image noise, and despite the shadow of the robot. The results from these experiments were more than good enough for navigation. Once again the vast majority of columns were interpreted correctly and with one exception errors were transient. The one exception was a particular stripe of red carpet in one corridor, which was uniformly considered an obstacle, at least when near the robot. This suggests that a richer representation of the image, perhaps using color rather than greyscale, might be more robust.

The previous work is described in detail by Martin [16], and forms the basis of the work reported here. Based on Martin's observations, we were interested in looking in more detail at the effects of the seed individual, and in the effects of training and testing in different environments.

3 Related Work

Several researchers have applied GP to finding things in images. The most directly related to the work proposed here is by Martin, and is described above. Also similar is work by Johnson [10], which uses GP to learn visual routines [19] that were used to determine human actions in the ALIVE virtual environment [15]. The learned routines ran over silhouette images of a human, and detected things like hand position. The system uses a specialized LISP-like language, similar to that used by Martin, and is capable of learning complex programs (on the order of hundreds of expressions).

Koza [12] used GP to learn programs for a simple optical character recognition task. More in-depth research

in the same domain was carried out by Andre [1]. Teller and Veloso used GP to learn programs that classify images in the PADO system [18].

Harvey, Husbands and Cliff [6] use GP to evolve the control and morphology of a very simple vision system for a gantry-based mobile robot. The system consists to three receptive fields and their position is learned by GP so that they can detect a small number of simple targets.

Graae, Nordin and Nordahl [5] evolve a program to calculate disparity from a pair of greyscale stereo images, using a sliding window representation, similar to that used by Martin.

Lutton and Martinez [14] investigate the use of GP to learn programs that extract geometric primitives from images. Interestingly, they found that the evolved programs were complementary to the widely-used Hough Transform [8]. For simple primitives, up to three parameters, the evolved programs perform poorly, and the Hough Transform is the method of choice. For more complex primitives, the memory and data requirements of the Hough Transform become unreasonable, but the evolved programs produce reasonable results.

Poli [17] uses a parse-tree based representation to evolve image filters. There are three terminals, all square blurring convolutions (2x3, 7x7 and 15x15), and six basic functions (add, subtract, multiply, divide, max and min). The learned programs were used to perform segmentation in medical images. They claim that the GP-based approach was more successful than neural network methods and “many other techniques reported in the image-analysis literature”.

Dumoulin *et al.* [4] use genetic algorithms to design image convolution operators that are implementable in reconfigurable hardware (FPGAs). Their program representation involves very simple bitwise operations and control flow, as one might expect when using FPGAs.

Not using genetic programming, but similar in spirit to the work proposed here, Draper, Bins and Baek [3] use reinforcement learning techniques to learn good sequences of image operators for detecting houses in aerial photographs. Other AI researchers have also applied planning techniques to induce good sequences of operators for image processing tasks [9].

4 Experiments

In this section, we describe the environments that we collected training and testing data from and outline

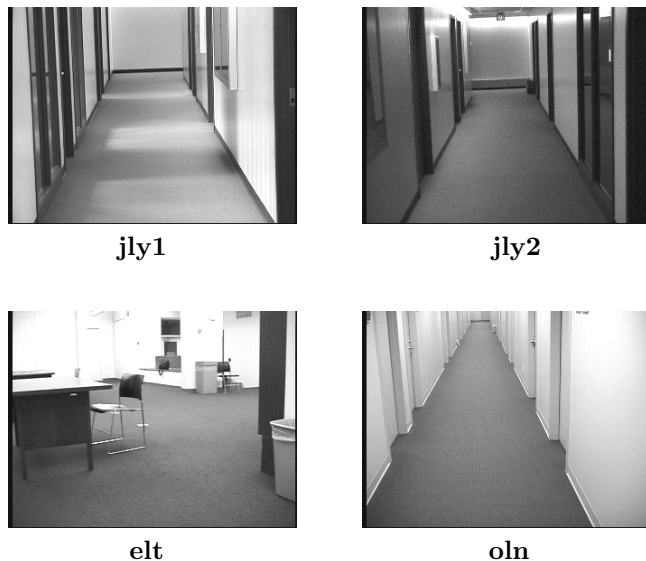


Figure 2: The four test environments. The checkered pattern evident in some images is a printing artifact and is not present in the actual environments.

our experimental setup.

4.1 Environments

We gathered data from four different indoor environments, examples of which are shown in figure 2. The details of these environments are as follows:

jly1 Office corridor environment with sharp shadows from strong sunlight coming through open doors and windows. White walls with dark molding at the bottom. Carpet is lighter than the molding, but darker than the walls.

jly2 Same environment as jly1, but without shadows.

elt Office conference room environment, with sharp shadows created by overhead lights and furniture. Light-colored walls, with dark carpet that is the same color as the molding.

oln Library environment, with no shadows. White walls, with white molding and darker carpet.

all Frames from all four data sets combined.

We took 350 frames of video while moving through each of these environments, and used every 5th frame to construct the training sets (each with 70 frames). Each of these frames has ground truth information, corresponding to the floor/wall interface in 6 pre-specified columns, added to it by a human. To evaluate fitness, we compare this ground truth information

for all of the images in a data set with the predicted positions, for a total of 420 fitness cases. The combined data set (**all**) uses a total of 70 frames, drawn uniformly from all four training data sets, and evaluates fitness in the same way as the four individual data sets.

4.2 Experimental Setup

Our setup exactly mirrors that of Martin’s second set of experiments (see section 2) where GP was used to find the lowest non-floor pixel in six pre-specified columns of an image. In this setup, only vertical iteration was used with the iterate node taking three (learned) arguments: the window size, horizontal location, and the sub-program to execute at every step. This setup produced individuals which achieved fitness levels exceeding 85% on Martin’s original training and test data, and similar levels on our own experiments. Fitness was evaluated as the sum of the absolute differences between the value predicted by the individual and human-supplied ground truth.

Our experiments used a population size of 4000. Each run consisted of a total of 51 generations, with individuals being chosen using tournament selection with a size of 7. Genetic operators include crossover (90% rate), with a 90% probability of selecting a function and a 10% probability of a terminal. Reproduction, or replication, also is used with entire, unchanged, individuals being moved to the next generation at a rate of 10%.

5 Experimental Results

We begin this section by discussing the performance of the GP system across the environments described in the previous section. During the course of running these experiments, we noticed an interesting bi-modal distribution in the final fitness of the best individuals. Either the best individual was very good or very bad. There were no “in-between” cases where the best individual had a middling performance. We discuss this further below.

5.1 Performance Across Environments

For each of the five training data sets, we performed 50 generations of GP, evaluating each generation on a single data set (the training set). At the end, we selected the best individual (on the training data), and evaluated it across the other data sets. The results for the cross-environment experiments are shown in table 1. Individuals evolved for a particular environment do

		Test Data Set				
Train	jly1	jly2	elt	oln	all	
jly1	81%	62%	36%	39%	56%	
jly2	63%	80%	21%	53%	57%	
elt	74%	74%	83%	74%	77%	
oln	12%	35%	39%	86%	40%	
all	78%	78%	77%	81%	79%	

Table 1: Performance of best individuals after 50 generations.

better there than in any of the others. Individuals do better in environments that are similar to those they were evolved in. Individuals trained on **jly1** and tested on **jly2** performed similarly to those trained on **jly2** and tested on **jly1**, suggesting that the best individuals learned to accommodate only one type of shadow, sharp or diffuse. Better performance was seen across environments that were similar (for example **jly1** and **jly2**) than across those that were significantly different (such as **jly1** and **elt**). Again, this is not surprising, and suggests that the best individuals are using features that are unique to their training environment. As expected, the results on the combined data sets are approximately the average of those on the individual ones.

One interesting feature of these results is that individuals tested on the **elt** data set uniformly perform badly (with the exception of those also trained on **elt**). However, individuals trained on **elt**, uniformly perform well on other test sets. The **elt** data set has less rigid structure and is more varied than the other environments (see figure 2). An individual trained in one of the other environments might exploit the corridor structure present there, since this will lead to better performance in the test set used during learning. However, when confronted with an environment where there is no such structure, the evolved individuals will perform very poorly. This suggests that less structured environments should be preferred for training, since they will tend to produce more general-purpose individuals. How to determine which environments meet this criterion, however, is not an easy problem.

Finally, individuals trained on a combination of all of the data performed well on all of the data sets. It is interesting to note that the performance of these individuals is not much better (in most cases) than the performance of the individuals trained on the **elt** data set. It seems that the difficulty and diversity of this training set is almost enough on its own to produce a robust algorithm that works across environments.

It is troubling, however, to see such over-fitting to such

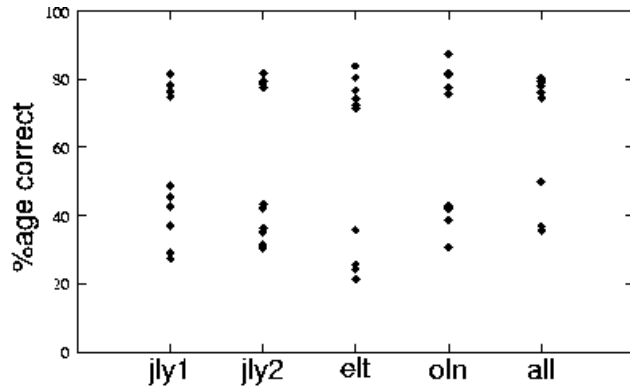


Figure 3: The bimodal distribution of the best individual performances.

a high degree in these data sets, as it leaves little hope that this would not occur in other problems. Yet, it can also be seen that this over-fitting can be largely avoided by making sure the training data has two important characteristics: diversity and difficulty. As displayed in the combined data set runs, an evolved individual will generalize much better if it has seen a wider variety of fitness cases. It follows that the more variety the individual has been exposed to, the better it will be at adapting to unseen environments. Difficulty plays an equal, if not more important, role in an individual’s adaptability. As demonstrated by **elt**’s performance in other environments, a training set containing difficult fitness cases forces the evolved algorithm to become more refined and consequently perform better in other environments where it is equally challenged.

5.2 Bimodal Performance Distribution

The best individual from each experiment fell into one of two distinct groups, one of high performance (70–90%), and one that failed to get very far past a fitness of 50%. Moreover, in all experiments there is a very definite point, over the space of a few generations, where the performance moves from the lower mode to the upper one. This can be clearly seen in figure 4. In all five of the data sets, if a fitness of 50% was not reached by generation 24 (in all experiments, except one that transitioned after generation 42), the run was doomed to remain in the lower mode. This grouping is very apparent in figure 3, which shows the performance of all individuals from one experimental run, for each of the data sets.

This behavior, where the fate of an experimental run is sealed by generation 24, suggests that an individual must learn some key operator, function, procedure, or

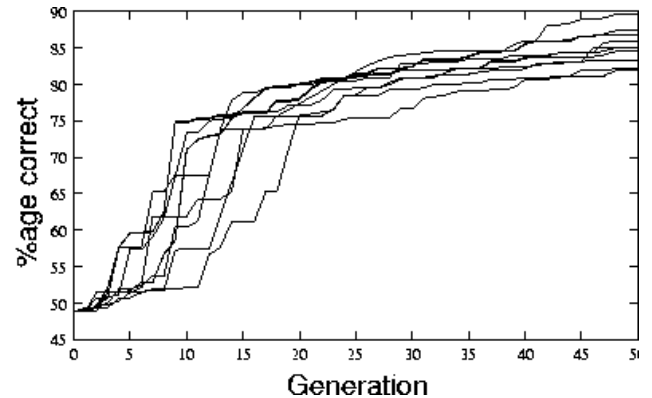


Figure 4: Ten best individuals from a seeded run in **jly1**

combination that is instrumental to its future success within those first 24 generations. If we can find out what this vital piece of the algorithm is, we could insert it into the population from the beginning. This should greatly increase the chances of a successful experiment, possibly even guaranteeing them.

In order to test this theorem we chose to examine the **jly1** data set. Since the learned programs are large, we chose the single best run from a particular experiment, and looked at how it changed with each generation. Most increases in performance are small, except between generations 16 and 17, where there was a 9% increase, and between generations 18 and 19, where there was an increase of approximately 7.5%. This seems to indicate that some important part of the feature detection program has been discovered at each of these points.

With our current tools, it proved extremely difficult to analyze the evolved individuals, and this is the focus of our current work. The changes in the best individual, even over a single generation, were often extensive, making it difficult to point to any particular feature as the cause of the improvement. Instead of identifying any particular key part of the individual, we instead re-ran the experiment, with the best individual from generation 19 in the initial population. All other initial individuals were created randomly. The results of the best individual from ten runs of the experiment are shown in figure 4. All parameters are the same as described above, and we ran each experiment for 50 generations.

In each of the ten experiments, the best individual performed well, with performances between 81.9% and 89.6%. The worst performance here, is similar to the *best* performance in the original set of experiments for this environment (see table 1).

The results of these new seeded experiments are more than simply running the experiment for 19 more generations. Martin’s original experiments were seeded with a poorly performing individual. This caused a significant performance increase. Although the same effect is observed in the our experiments, the *quality* of the seed individual is radically different. Martin’s seed was a hand-coded program and, as such, was relatively compact and efficient. The seed in our experiments was an evolved individual, which was all but unintelligible to a human programmer. Although it correctly identified about half of the floor/wall interfaces, it did not do it in the most efficient manner, and it is bound to have redundant code.

It is this very redundancy that gives it an advantage over Martin’s seed. Compact code cannot be altered much without breaking it. This makes it brittle when subjected to GP operations. The performance of the best individual will be unlikely to exceed that of the seed because of this. Any changes tried will make the seed code perform worse. Conversely, because of the redundancy of the evolved seed, non-catastrophic changes *can* be made to it. Because of its size, there are many opportunities for crossover to select a working sub-program. We believe that these reasons make the use of a less finely-tuned seed a better option, at least in the domain in which we performed these experiments.

6 Conclusions and Future Work

In this paper, we have presented the initial results of using GP to learn a visual feature detector for mobile robot navigation domains. We discussed the results of experiments across several environments, and analyzed a bi-modal learning behavior, which resulted in the final performance either being very good or very bad.

Although this paper deals with a particular set of experiments in a particular domain, we believe that it sheds some light on GP in general. Although these observations are certainly not new, our experiments give a simple example of the principles in action.

The choice of an appropriate seed for the population seems to be crucial, and can have a huge effect on the final performance level of the system. There is a hidden danger in choosing a seed that is too mature, compact and efficient. Such a seed can dominate the population, and is not amenable to incremental improvement. This will doom the system to perform at or below the level of the original seed individual, unless better a better performing individual can be learned from scratch. In such a case, however, we believe that

the inclusion of a “perfect” seed will actually slow the learning of this better performing individual. One of the things that we plan to investigate in future work is the question of how the quality (in terms of size, efficiency, *etc.*) of the initial seeds affects the final performance of the learned individuals.

We have also begun to analyze the individuals in order to identify subtrees, or terminal and function clusters that seem to be largely responsible for large performance increases over a small number of generations. By turning these into functions, and adding them into the pool of terminals that the GP system has access to, we hope to increase the speed of learning over similar environments and tasks, without stifling the learning process with whole seeds, as described above. An interesting offshoot of this work will be to see if GP finds the same sorts of image operations, or combinations of operations that a human expert would use, or if it comes up with radically different solutions.

The research done here also supports the idea of having difficult and diverse training data so that the evolved individuals generalize well to other applications. The performance of the combined data set supports the common sense notion that the greater variety of fitness cases the evolved individual encounters, the better it will perform in a variety of situations. The importance of difficult training data also was highlighted in that a difficult data set forces GP to produce a more robust solution. This will, in turn, tend to be more robust across changes in environment (within reasonable limits). We also plan to experiment with the use of co-evolution techniques, in order to maintain the difficulty and diversity of the training data throughout the experimental run.

References

- [1] David Andre. Automatically defined features: The simultaneous evolution of 2-dimensional feature detectors and an algorithm for using them. In Kenneth E. Kinnear Jr., editor, *Advances in Genetic Programming*, chapter 23. MIT Press, Cambridge, MA, 1994.
- [2] Stefano Cagnoni, Riccardo Poli, George Smith, David Corne, Martin Oates, Emma Hart, Pier Luca Lanzi, Egbert Jan Willem, Yun Li, Ben Paechter, and Terence C. Fogarty, editors. *Real-World Applications of Evolutionary Computing*, volume 1803 of *Lecture Notes in Computer Science*. Springer, Berlin, Germany, 2000.

- [3] Bruce A. Draper, Jose Bins, and Kyungim Baek. ADORE: Adaptive object recognition. *Videre: Journal of Computer Vision Research*, 1(4), 2000.
- [4] Joe Dumoulin, James A. Foster, James F. Frenzel, and Steve McGrew. Special purpose image convolution with evolvable hardware. In Cagnoni et al. [2], pages 1–11.
- [5] Christopher T. M. Graae, Peter Nordin, and Mats Nordahl. Stereoscopic vision for a humanoid robot using genetic programming. In Cagnoni et al. [2], pages 12–21.
- [6] Inman Harvey, Phil Husbands, and Dave Cliff. Seeing the light: Artificial evolution, real vision. In Dave Cliff, Phil Husbands, Jean-Arcady Meyer, and Stuart W. Wilson, editors, *From Animals to Animats: Proceedings of the Third International Conference on the Simulation of Adaptive Behavior*, pages 392–401, Cambridge, MA, 1994. MIT Press.
- [7] Ian Horswill. Polly: A vision-based artificial agent. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, 1993.
- [8] P. V. H. Hough. A new method and means for recognizing complex pattern, 1962. U.S. Patent 30690653.
- [9] X. Jiang and H. Binke. Vision planner for an intelligent multisensory vision system. In *Automatic Object Recognition IV*, pages 226–237. 1994.
- [10] Michael Patrick Johnson. Evolving visual routines. Master’s thesis, Massachusetts Institute of Technology, Cambridge, MA, 1995.
- [11] John Koza. *Genetic Programming*. MIT Press, Cambridge, MA, 1992.
- [12] John Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA, 1994.
- [13] John R. Koza, Forrest H. Bennett III, David Andre, and Martin A. Keane. *Genetic Programming III: Automatic Programming and Automatic Circuit Synthesis*. Morgan Kaufmann, San Francisco, CA, 1999.
- [14] Evelyne Lutton and Patrice Marintez. A genetic algorithm with sharing for the detection of 2d geometric primitives in images. In Jean-Marc Alliot, Evelyne Lutton, Edmund Ronald, and Dominique Snyers, editors, *Artificial Evolution*, volume 1063 of *Lecture Notes in Computer Science*, pages 287–303. Springer, Berlin, Germany, 1995.
- [15] Pattie Maes, Trevor Darrell, Bruce Blumberg, and Sandy Pentland. The ALIVE system: Full-body interaction with animated autonomous agents. In *Proceedings of the Computer Animation Conference, Geneva, Switzerland*. IEEE Press, 1995.
- [16] Martin C. Martin. *The Simulated Evolution of Robot Perception*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2001.
- [17] Riccardo Poli. Genetic programming for feature detection and image segmentation. In Terence C. Fogarty, editor, *Evolutionary Computing*, volume 1143 of *Lecture Notes in Computer Science*, pages 110–125. Springer, Berlin, Germany, 1996.
- [18] Astro Teller and Manuella Veloso. PADO: Learning tree-structured algorithms for orchestration into an object recognition system. Technical Report CMU-CS-95-101, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1995.
- [19] Shimon Ullman. Visual routines. *Cognition*, 18:97–159, 1984.