**ENEL420**

Signals Assignment 2 - Progress Report

Matt Kokshoorn

Nick Bingham

23rd August 201

# GENETIC ALGORITHMS BACKGROUND

Engineering is a process of trying to find an optimal solution for a given problem. A genetic algorithm emulates natural selection to try and iteratively determine a solution to a defined problem which is very close to the ideal solution. It is one of the class of evolutionary algorithms, which all incorporate genetic principles such as inheritance, mutation, selection and crossover.

Genetic algorithms work by taking a population of possible solutions (referred to as individuals or phenotypes) and evolving these into a better solution. Each population of solutions is a generation. The first generation is usually randomly generated, although it may be seeded in close proximity to an optimal solution. Subsequent generations are then created by taking a selection of the fittest solutions in the generation and using a combination of crossover (breeding) and/or mutation to produce new solutions. The fitness of each solution is determined by quantifying a range of output factors to determine how close its properties are to ideal. Once a certain predefined fitness level is met or a predefined number of generations have been analysed the algorithm will terminate and present the best solution.

There are two major requirements that need to be met so that a genetic algorithm can solve a given problem. First, the problem must be capable of genetic representation. This requires that values and variables are capable of being stored and mutated in a meaningful way. Second, a fitness function must exist such that each solution can be graded to allow selection of the optimal results.

A genetic algorithm has four stages of operation. First, the initial generation needs to be generated. This is normally random. Second, the top solutions in the generation need to be selected. Third, there must be some modifications to these top solutions (through genetic crossover and mutation) to generate the following generation. The second and third steps then need to be repeated. Finally, there must be termination when the design constraints are suitably met.

There are several downsides to genetic algorithms. They are computationally intense, and do not work well for complex systems. They best solutions have limited creativity, as they are only the best solution given the fitness function. These algorithms also tend to converge on local maxima, although this can be avoided by imposing penalties for similarity in solutions.

# IMAGE PROCESSING APPLICATION

**Edge Detection**Within the field of digital image processing edge detection is an important tool. An ‘edge’ is considered to be a sharp change in value of an image. Depending on the representation of this each value, this will be a change in colour, intensity or both. A series of these edge points can be identified in a 2D image to form a line or grouped together to form a feature. Edge detection has many applications in computer vision, perhaps the most important of these being feature detection.

As previously mentioned a group of unique edge points can group together to represent a feature. Depending on the uniqueness of this feature, its location can usually be found in another image of a similar environment or a time delayed image of the same environment. The location of such features can then be used in computer vision applications such as object recognition and calculating the optical flow.  
  
There exist a number of edge detection methods that all have their associated advantages and disadvantages. Most edge detectors use a gradient operator that can be realised through an N by N Matrix or ‘Kernel’. This can then be convolved with an image in both a Y and X direction to produce an image whose element values correspond to the strength of the edge. Some common edge detectors are explained below.

**Canny Edge Detector**The canny edge detector is a multi-stage algorithm .First the 2nd derivative or Laplacian of a Gaussian operator or equivalent kernel must be obtained. This is then convolved with the image in both directions. The result is an image representing the gradients of the original image. This can then be normalised, thresholded and thinned to achieve a set of edge points in image.

**Frequency Domain Edge Detection**Another method of edge detection is to take the Fourier transform of an image and then remove the low frequencies from the image. In the frequency domain these low frequencies are omitted by removing a small circle around the centre of the image. This operates on the principal that edges correspond to sharp changes in intensity. By then taking the inverse Fourier transform of the image the edges points of the image can be seen.

**Kernel Approximations**The methods previously described, among others can often be broken down to or approximated by convolution of a pair of kernel operators often followed by normalisation, thresholding and thinning. Some commonly used operators include the; Roberts Cross Operator, 3x3 Prewitt Operator, Sobel operator and a 4x4 Prewitt Operator shown below Figure 1.



Figure – Common Gradient Operators:   
a) Roberts Cross Operator b) 3x3 Prewitt Operator c) Sobel Operator and d) 4x4 Prewitt Operator

**Trade-offs and Performance Measures**An optimal edge detector is described to have three key properties; good detection, good localisation and minimal response. Good detection refers to the general detection performance in that the algorithm should reveal as many real edged as possible. Localization refers to the edges being as marked as close as possible to their edge in the original image. Minimal response refers the edges being only marked once and concerns the detectors immunity to noise in the image.

Furthermore another key performance parameter is the size of the kernel operator. Large kernel sizes allow the edge detector to detect large-scale edges within the image whereas smaller kernel sizes allow finer features to be detected within the image. Other factors also affect the performance of the edge detector such as the thresholding value and thinning. Some of the edge detection methods previously discussed can be seen below in Figure 1.



Figure - Comparison of a Range of Edge Detectors.   
From Left to Right: Original Image, Laplacian Method, Sobel Filter and Canny Edge Detection [1].

# PLAN

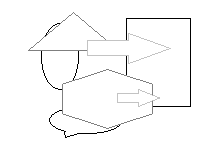
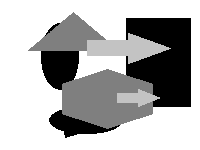
The aim of this project is to use a genetic algorithm to determine an optimal edge detection method. In order to do this, there needs to be a genetic representation of possible solutions and a fitness function. As one matrix/kernel can be expressed as a transform of the other i.e. the y convolution matrix and transform of the x direction matrix only one of the two need to be considered. For the purpose of the below diagram the x direction kernel of the Sobel operator has been used and a thresholding value of eight.

Figure - Example of Genetic Representation of Sobel Operator and Threshold Value.

All edge detection methods can be represented as two operation matrices. It is straightforward to construct a genetic representation of an edge detection method where these matrices are kept as a fixed size. A 1D number array can be made directly from the matrices which make crossover and mutation trivial to carry out, see Figure 3. However, if these matrices are variable length crossover is far more difficult. One aspect of the project will be looking into crossover of different size matrices to see if a novel method of doing this can be determined.

There are a range of performance indicators which, by appropriate weighting and combining, can be used to calculate a fitness value for a given solution. Processing requirements can be measured, with a lower processing time being preferred. Similarity of the solution output image to the ideal output image can be measured by finding the point of highest correlation, where a higher correlation value at this point is preferred. The displacement of this point from the ideal image is also a performance measure, with no displacement being optimal. Any image repetition can be measured using other points of high correlation, with less image repetition being preferred.

In order to construct an optimal image, an image of only edges will be created. This can then be filled in appropriately to generate the input image. The only-edge image can then be compared to outputs of edge detection attempts to determine how good they are. Another important performance measure of an edge detector is its robustness to noise. To factor this in to the fitness fucntion similar images to those shown below will also have noise introduced to them and again be compared to the ideal output.



# REFERENCES

[1] Lee, M. K., Leung, S. W., Pun, T. L., Cheung, H. L., & Lee, A. M. (2000). Edge detection by genetic algorithm. In *Image Processing, 2000. Proceedings. 2000 International Conference on* (Vol. 1, pp. 478-480). IEEE.