Image Compression of Natural Images Using Artificial Gene Regulatory Networks

Martin A. Trefzer
University of York
YO10 5DD, York, UK
mt540@ohm.york.ac.uk

Tuze Kuyucu
University of York
YO10 5DD, York, UK
tk519@ohm.york.ac.uk

Julian F. Miller
University of York
YO10 5DD, York, UK
jfm7@ohm.york.ac.uk

Andy M. Tyrrell
University of York
YO10 5DD, York, UK
amt@ohm.york.ac.uk

ABSTRACT

A novel approach to image compression using a gene regulatory network (GRN) based artificial developmental system (ADS) is introduced. The proposed algorithm exploits the fact that a series of complex organisms ($\equiv$ developmental states) can be represented via a GRN description and the indices of the developmental steps in which they occur. Organisms are interpreted as tiles of an image at each developmental step which results in the (re-)construction of an image during the developmental process. It is shown that GRNs are suitable for image compression and achieve higher compression rates than JPEG when optimised for a particular image. It is also shown that the same GRN has the potential to encode multiple images, each represented by a different series of numbers of developmental steps.

Track: Generative and Developmental Systems (GDS).

Categories and Subject Descriptors

I.2.2 [**Computing Methodologies**]: Artificial Intelligence—*Automatic Programming*; I.4.2 [**Image Processing and Computer Vision**]: Compression (Coding)—*Approximate methods*

General Terms

Algorithms

1. INTRODUCTION

One of the most fascinating properties of biological development is the fact that large, complex organisms are created using the relatively small amount of information that is encoded in the deoxyribonucleic acid (DNA) of a single cell. Once development is initiated, a complex process of gene activity and regulation takes place that triggers a series of cell actions which, over time, create complex, multicellular organisms [18]. During the organism's lifetime, the developmental process constantly adapts to changing environments and maintains it in the case of damage. Hence, the crucial factor that allows developmental processes to unfold their power is time. From an engineering and computer science point of view, this means that information is encoded in the time steps of a dynamic system. In this respect, the DNA represents compressed information, which is expanded during the developmental process and is expressed through the various states of the organism.

Most of the current research and applications in the area of artificial developmental systems (ADS) concentrate on the properties of development that also help biological organisms to survive and maintain themselves, namely adaptivity, robustness and self repair. In addition to scalability these features are also desirable in engineered systems like embedded systems, robots and control systems [7, 2, 3, 14, 13]. One of the most prominent applications for studying ADSs in the field of evolutionary computation (EC) and biological modeling is pattern generation and pattern formation [12, 4, 14, 3], which is also the inspiration for the algorithm introduced in this paper. In most of the work, a two-dimensional array of identical entities (cells) is used to achieve a 2D pattern which is defined through the cell states or protein concentrations. There are examples where development takes place in three dimensions and examples where cells have different shapes and can migrate. Questions which are addressed in pattern formation experiments include growth, symmetry, shaping, stability and recovery from damage [4, 14].

In contrast the idea behind this work is to optimise a GRN in a manner such that it matches patterns of a given set of target patterns as closely as possible at arbitrary developmental steps and use it for image compression. The use of a GRN based ADS is motivated by the proven ability of such systems to create a variety of patterns and structures, and by the fact that ADSs represent dynamic systems, which can be optimised to encode useful information in their state space. Furthermore, GRN based implementations have the potential for hardware and embedded systems friendly implementation (see Section 3). For a defined starting point, the developed organisms can be enumerated according to the order in which they occur while running the ADS. Hence, a sequence of complex states can be repre-

GECCO'10, July 7–11, 2010, Portland, Oregon, USA.

sented by the GRN and a sequence of integers. If the pattern at each developmental step is interpreted as a tile of a larger image, the image will be (re-)constructed during the developmental process. From a data compression point of view this is ideal since the states can become arbitrarily complex while the amount of compressed data remains constant for a fixed number of developmental steps. However, there are two major issues that render this kind of data compression impractical: first, optimisation of a dynamic system is time consuming and requires a large amount of computation. Second, it is not necessarily guaranteed that an arbitrary set of data can be exactly matched by any implementation of an ADS. While dynamic systems are probably not practical for lossless data compression, this work investigates the application of GRNs to lossy image data compression. In the latter case, it is not necessary to encode a perfect version of the input data, as long as sufficient image quality is achieved. In cases where smaller amounts of data are more important than maximum quality, the compression rate can be increased at the cost of losing information, i.e. losing image detail.

The results in this paper show that GRNs are suitable for image compression and achieve a higher compression rate than JPEG when optimised for a particular image. The behaviour of the dynamic system is investigated by visualising the output tile space of the developmental process, which provides insight on how the ADS explores the feature space of the image. Furthermore it is shown that the same GRN has the potential to encode multiple images, each represented by different series of numbers of developmental steps. Example images (including SIPI-IDs) *Lena (4.2.04)*, *Elaine (elaine.512)*, *Mandrill (4.2.03)* and *Aerial01 (2.1.01)* are taken from the USC Signal and Image Processing Institute database [17] and are originally encoded in the lossless Tagged Image File Format (TIFF). The *Lena* image is greyscaled using the GNU Image Manipulation Program (GIMP, http://www.gimp.org/), version 2.6.6. Images which are encoded using the proposed algorithm are denoted as Gene Compressed Image (GCI), which is also proposed as the name and file extension for the image format.

2. IMAGE COMPRESSION

Joint Photographic Experts Group (JPEG) image compression and image compression via vector quantisation (VQ) are relevant for the proposed image compression algorithm and the results presented. JPEG compression is chosen as a benchmark for the reason that JPEG is a widely adopted image compression standard for natural images. Hence, the assessment of the proposed algorithm can be based on a method which is established and optimised.

2.1 JPEG Image Compression Standard

JPEG [5] is a lossy image compression algorithm. The encoding process encompasses the following steps: first, the colour space of the image is converted into YCbCr, where Y is the luma component—which effectively represents an 8bit grey scale version of the image—and Cb and Cr represent the colours. Second, the resolution of the colour channels is reduced, usually by a factor 2. This is called sub-sampling and exploits the fact that the human eye has a lower resolution for colour than for brightness. Third, the image is split into tiles of 8×8 pixels. Each of those tiles is converted to the frequency domain using a two-dimensional discrete cosine transform (DCT). The fifth step achieves the actual data reduction (quantisation), by dividing each of the amplitude values by the corresponding value in a quantisation matrix. The quantisation matrix is the same for all tiles and predefined. Again, the fact that the human eye is less sensitive to small localised variations in colour or brightness (small details) is exploited by storing the high frequency components with a lower accuracy. The lower the quality setting, the more of the high-frequency components are completely discarded. Finally, the resulting data is compressed using a lossless entropy encoder, i.e. Huffman encoding [1].

In order to decode the image these steps are performed in reverse order. Quantisation is reversed by multiplying each component with the corresponding entry of the quantisation matrix, which is stored within the JPEG file. Quantisation in JPEG is the reason why it is a lossy compression method, where the amount of data lost is determined by the values of the matrix elements of the quantisation matrix. For the DCT there exists an inverse transformation (DCT^{-1}), which is used there to convert back to the time domain.

2.2 Vector Quantisation

Image compression via VQ is based on the idea that a series of input vectors, which correspond to parts of the original image, can be represented by an index that points to the closest match within a minimised codebook, thereby reducing the amount of data that needs to be stored. At the encoding (compression) stage, each input vector is matched against a codebook and represented by the index of the codeword with the smallest euclidean distance to the input. At the decoding (decompression) stage, the original data is reconstructed by combining the codewords from the codebook according to a sequence of indices. The smaller the size of the codebook the higher the compression rate at the cost of increased data loss. In the case of the proposed method, the codebook is represented by the GRN which creates the codewords during the developmental process. This has the advantage that the 'codebook' does not change in size when generating a larger number of tiles, since development could simply be run for longer, exploiting the fact that information is encoded in time, rather than additional code words. In VQ there are several algorithms to find and optimise the codebook, for instance, pairwise nearest neighbour (PNN), simulated annealing, maximum descent (MD), and frequency-sensitive competitive learning (FSCL) and Linde-Buzo-Gray (LBG) [11, 19]. There are also approaches to optimise the codebook via EC [10, 15]. Like optimising dynamic systems (DS), finding the optimal codebook is an NP-hard problem and may take a long time when high quality is required, which is the reason why VQ is not more widely used.

3. THE DEVELOPMENTAL SYSTEM

The GRN based model for artificial development used in this paper is based on the one that has been introduced in [16, 8]. The design considerations of the original ADS are retained, namely the usage of data structures and operations in the GRN core that are suitable for embedded systems, i.e. boolean, integers, no division and to keep the mechanisms of the ADS as close as possible to their biological counterparts within the boundaries of the chosen data types. Whilst not crucial for the experiments in this paper, the choice of the data structures imposes no loss of generality and is therefore unchanged. However, some improvements are made to

Algorithm 1 The function of the structuring protein is given in pseudo-code. Structuring gene actions express pixel values or frequency components of a rectangular image area. When a gene is expressed, the functions of all of its structuring proteins are carried out and modify the cell type (=current pixel value/frequency component).

par_1, par_2 = parameters of structuring protein
ctype = cell type (pixel value, freq. component)
n_{chem} = total no of protein and molecule types
chem = protein or molecule type
lev(chem) = chemical level

chem $\leftarrow$ par_2 mod #chem

switch $par_1 \equiv 0 (\text{mod } 16)$
  case 0 **:** ctype = lev(chem)
  case 1 **:** ctype = -lev(chem)
  case 2 **:** ctype = -ctype
  case 3 **:** ctype = lev(chem)-lev((chem+1) mod n_{chem})
  case 4 **:** ctype = lev(chem)-lev((chem+2) mod n_{chem})
  case 5 **:** ctype = lev(chem)-lev((chem+3) mod n_{chem})
  case 6 **:** ctype = lev((chem+1) mod n_{chem})-lev(chem)
  case 7 **:** ctype = lev((chem+2) mod n_{chem})-lev(chem)
  case 8 **:** ctype = lev((chem+3) mod n_{chem})-lev(chem)
  case 9 **:** ctype = ctype+7;
  case 10 **:** ctype = ctype-7;
  case 11 **:** ctype = ctype+4;
  case 12 **:** ctype = ctype-4;
  case 13 **:** ctype = ctype+1;
  case 14 **:** ctype = ctype-1;
  case 15 **:** ctype = ctype;
end switch

the ADS for this paper: first, a dedicated diffusion layer is added and only chemicals that are released to this layer by the cells are subject to diffusion. Chemicals need to be absorbed by the cells from the diffusion layer first, before they affect gene regulation. This is motivated by natural development. Second, a genetic representation that allows for variable length GRNs is used in this paper, which allows for a more flexible and compact encoding of the genes, see Figure 2. An overview of the mechanisms of the ADS is provided in the following sections. The term chemicals refers to both proteins and molecules.

3.1 Representation and Gene Regulation

The core of the developmental model is represented by a GRN, as shown in Figure 1. The DNA is implemented as a string of symbols that encode start and end of genes, separation of pre- and postcondition within genes, binding sites and chemicals as shown in Figure 2. Genes interact through chemicals and form a regulatory network. There is one major difference between the artificial model and biology: in the ADS used the binding sites match exactly one chemical, whereas in natural genes binding sites are defined by certain upstream and downstream gene sequences that accept a number of proteins to bind and transcribe their genetic code. The binding sites in natural DNA therefore allow for smooth binding, i.e. the probability that certain chemicals (transcription factors) bind to the DNA is given by how well the binding sites of the chemical matches the one of the DNA. There are examples of artificial GRNs where soft-binding is included [6, 4]. However, it is not clear whether including soft-binding yields major benefits and the inclusion of this feature is extremely expensive in terms of memory requirement and computation effort.

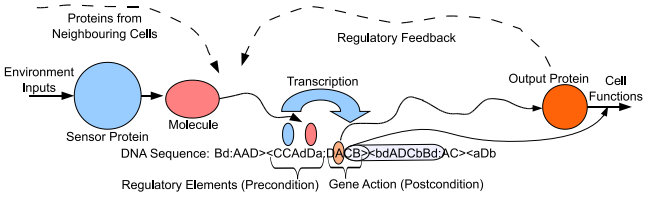


Figure 1: The genes in a DNA sequence are activated when a sufficient amount of protein is produced by the GRN, which then causes transcription of the respective gene. In the example shown in the figure, the activity of the GRN results in the correct transcription factors (the promoter proteins) that bind to the sites of a gene sequence. This initiates the production of another protein, which can then affect the cell functionality using further information that is encoded in the gene. The produced protein also provides feedback to the GRN, which regulates the transcription of further genes, hence, dynamic gene regulation is achieved. Note that when any inhibitory condition is met, the gene is not expressed.

In principle the ADS used can work with an arbitrary number of chemicals. However, the current system is working with eight chemicals: four proteins and four molecules. Gene regulation is described in Figure 2. A description of the proteins and their roles is given in table 1. The total number of chemicals is arbitrarily chosen. Increasing the number of chemicals increases the complexity of the dynamics of ADS, but makes it slower to process. The presence of molecules, which cannot be directly produced by the GRN, motivated by natural development where they play a significant role in gene regulation [18].

3.2 Evolution of the DNA

The DNA is derived from a genome that is evolved using a standard 1+4 evolutionary strategy (ES). The EA parameters are based on values that are widely used in the field of evolutionary computation (EC). The genome is represented by a string of integers and mutation takes place by replacing them with a new random value at a rate of 2% of the genome length. Crossover is not used. The GRN is obtained by mapping the string of integers to GRN symbols using the modulus operation on the genome in a straight forward fashion. Variable length genes are achieved via active/inactive flags encoded in the genes.

3.3 Cell Communication and Growth

Wolpert [18] describes three different types of cell signalling: chemical diffusion, direct contact of complementary chemicals on the cell's surface, and gap junctions (Plasmodesmata). Two different types of cell signalling are realised in the proposed model. Chemical diffusion has been implemented in a similar way as it takes place in physical systems, e.g. a drop of ink dissolving in water. Half of the cells chemical is thereby distributed amongst its nearest neighbours in equal shares, i.e. each neighbour obtains $\frac{1}{8}$ of the cell's chemical. Chemical levels are credited and debited after the GRN has processed the next developmental step for all

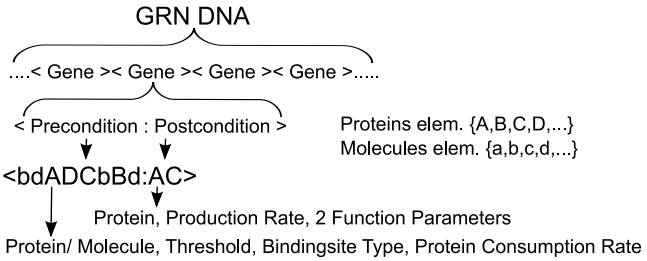


Figure 2: An example DNA is shown. The GRN comprises a number of genes that are separated by brackets. A gene consists of pre- and postcondition, which are separated by a colon. Each letter refers to one type of chemical and three integer parameters. Both proteins and molecules are present in the gene's precondition and contribute to gene regulation whereas only proteins can be directly produced in the postcondition. In the case of precondition, the letter defines which type of chemical can bind to this site. The parameters are interpreted as the protein threshold that is necessary to activate the binding site, the type of binding site (inhibitory or excitatory), and the rate at which the binding protein is consumed while the site is active. In the case of postcondition, the parameters are interpreted as the production rate of the respective protein and as parameters for the respective protein function. An overview of the protein functions is shown in table 1.

cells. Thus, the GRN always works with the original chemical levels and the order of cell update should therefore not bias the course of development. Diffusion only takes place on a separate diffusion layer which is underlying the array of cells (organism). Only chemicals that are released by a cell via the according cell action participate in the diffusion process. Vice versa, cells can absorb chemicals from the diffusion layer, which only then affect gene regulation within the respective cell. Diffusion is a signalling mechanism that helps maintaining symmetries within the system.

The second cell signalling mechanism is an implementation of Plasmodesmata in plants (protein tunnels) [9]. The GRN produces a Plasmodesma protein for one of the four directions N,S,E,W. If the neighbour cell is alive and produces the complementary Plasmodesma protein, a tunnel will be established between the two cells with the effect that all chemicals are equally shared between those cells, as depicted in Figure 3. This tunnel is active as long as the necessary Plasmodesma proteins are produced. Complementary to diffusion, the tunnelling mechanism enables the GRN to break symmetries within the system, due to the fact that tunnels do not necessarily occur on all four sides of the cell at the same time.

Cell growth is achieved using the Plasmodesma proteins. If the neighbour cell, which is targeted by the tunnel, is an empty or dead cell, it will become alive during the next developmental step (see Figure 3). Cell death is not implemented at the current stage.

The cells are surrounded by border cells, which absorb any amount of chemical that diffuses outwards on the diffusion layer. Since the border cells are inactive, i.e. not processed by the GRN, they are unable to produce chemicals and form tunnels. Hence, tunneling outside the bounds of the organism is impossible.

Table 1: The current GRN works with four proteins and four molecules. Proteins are directly produced by the GRN, whereas molecules are only a product of a gene function as a result of a measurement or interaction that is performed by a protein. Their roles in development are described in the table. Please note that all proteins and messenger molecules take part in gene regulation, in addition to their special purpose.

Protein/ Molecule	Role (apart from gene regulation)
Structuring (Protein A)	Structuring proteins are used to alter the cell state according to a set of predefined rules. In this paper, the cell state represents a pixel value in the time-domain or a coefficient in the frequency-domain. The structuring function is described in Algorithm 1.
Sensory/ Interacting (Protein B)	Sensory proteins provide a means to react to changes in the cell state. Molecules are produced, based on the current cell function, cell location and possible rules that can be encoded in the genes.
Diffusion Layer (Protein C)	Diffusion layer proteins release chemicals to the diffusion layer and absorb them from the diffusion layer. Only chemicals which are released to the diffusion layer are subject to the diffusion process. Chemicals on the diffusion layer need to be absorbed by the cell first, before they affect gene regulation.
Plasmodesma (Protein D)	Plasmodesma proteins provide a mechanism to form Plasmodesmata [9] in order to share their proteins with neighbour cells, and they initiate growth.
Molecules (a,b,c,d)	Molecules are produced as a result of the sensor protein being expressed and reacting according to the current cell state or environment input.

3.4 Developing Organisms to Form Images

The application in this paper is image compression via GRN. Therefore, the GRN has to be able to represent an image or parts of an image. This is achieved by designing the function of the structuring protein in such a manner so that it manipulates an integer value, which represents the cell state. In one developmental step the cell state is therefore changed multiple times according to the number of structuring proteins produced. It is then possible to interpret the states of a rectangular area of cells as a corresponding area of pixel values where one cell represents one pixel. In the time domain it would be sufficient to cover a value range of $0 \dots 255$ in order to represent an 8bit grey-scale image. However, for many image processing applications it is more beneficial to represent images in the frequency domain using DCT or Fourier transformation. In both cases the rectangu-

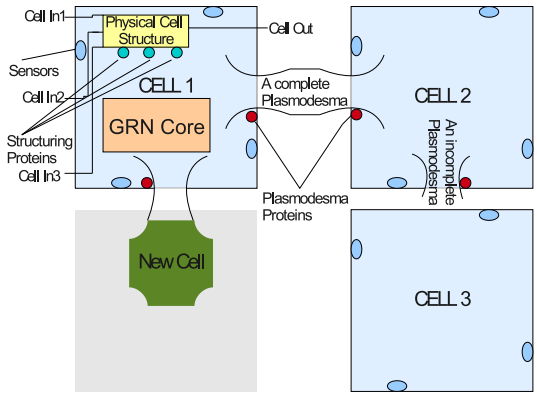


Figure 3: In a multi-cellular environment using the four protein types from Table 1 a cell is able to: interact with its environment, multiply, represent a pixel value or frequency component, and form a complex multicellular organism.

lar array of pixel values becomes an array of coefficients for two-dimensional frequency components of the image. Usually, the upper left element represents the DC coefficient and the frequency increases towards the lower right coefficient. The coefficients can take on relatively large positive and negative values (± 800 when applying DCT to *Lena*). Hence the value range of the cell state has to be wider than $0 \ldots 255$ and also cover negative values. Due to the fact that chemical levels can only take on positive values, a mapping function is required that is able to compute the required value range based on the output of the ADS. As can be seen from the description of the mapping function in Algorithm 1, a relatively simple and straight forward implementation is used in this paper. This suggests that there might be scope for further improving the performance of the proposed method at this point.

4. IMAGE COMPRESSION USING GRN

The presented image compression algorithm is based on the idea of replacing 8×8 pixel tiles (64 bytes in the time domain or 64 integers in the frequency domain) of an image with a single integer value in order to achieve data compression. This is similar to VQ 2.2, where input tiles of an image are represented by the index of a tile from a codebook that offers the least distortion. Hence, in order to restore the image in VQ, only the codebook and the array of indices obtained from encoding the input vectors need to be saved. The smaller the codebook is compared to the input space, the greater is the data reduction but also the loss of information.

In this paper a GRN based ADS is used, which is able to produce a series of 8×8 cell organisms during the developmental process. For encoding, an EA is used to optimise the GRN to produce organisms that are able to represent the 8×8 pixel tiles of an image in such a way that for each one the distortion is minimised at some developmental step. The encoded image is then defined by the GRN and the sequence of the numbers of the developmental steps where the closest matches of the input tiles occur. Note that the latter sequence is in the order in which the tiles occur in the image, rather than in the order the ADS develops them.

Algorithm 2 GRN based image encoding (compression) and decoding (decompression) is described in pseudo-code. Multi-pass encoding can be achieved by repeatedly running the *encoder* and *decoder*. The term GCI file refers to genetically compressed image (GCI) files.

Encoder:

1. Divide input image into a set of 8×8 tiles
2. Convert tiles to frequency domain using DCT {optional}
3. Optimise GRN to produce a stream of organisms (tiles) with minimum distortion to the input tiles via EA
4. Initialise development:
   Initialise all chemical levels to 0
   Initialise 8×8 cells organism
   Set cell(1,1) alive
5. During development:
   Record iteration numbers of organisms with minimum distortion to input tiles
6. Save GRN and iteration numbers to GCI file
7. Use bzip2 for entropy encoding of GCI file

Decoder:

1. Load and unzip GCI file
2. Initialise development as in *Encoder*
3. Run loaded GRN
4. During development:
   When iteration number matches one from the GCI file replace it with the current organism
5. Convert tiles to time domain using DCT^{-1} {if done for encoding}
6. Reconstruct image

Multi-pass:

1. Subtract the resulting image tiles from the ones of the original image (or the previous pass)
2. Run the *Encoder* on the resulting tiles
3. For decoding, run the *Decoder* using the iteration numbers and GRNs from each pass separately
4. Add up all resulting sets of tiles and reconstruct the image

In this respect the GRN and iteration numbers work similar to the codebook and indices in VQ. In order to decode an image, the developmental process is (re-)run using the optimised GRN, which reproduces the series of 8×8 cell organisms (image tiles). The image is then restored by arranging tiles according to the sequence of iteration numbers. Since the optimisation of the GRN is an NP-hard problem, the encoding process is time consuming (2-3 days for high quality), whereas the decoding process is fast (seconds). The compression algorithm is described in Algorithm 2. Fitness is computed as the sum of mean square errors of the tiles of the original image and the best matching ones that are encountered during developed.

It is further possible to perform multi-pass encoding on the input image, which results in a larger GCI file since for each pass another GRN and set of iteration numbers needs to be stored. However as shown in the results, multi-pass encoding can both speed up the encoding and achieving higher quality of the compressed images. A tile size of 8×8 pixels of grey-scaled images is used for the following reasons: first, 8×8 pixel tiles are widely used in image compression

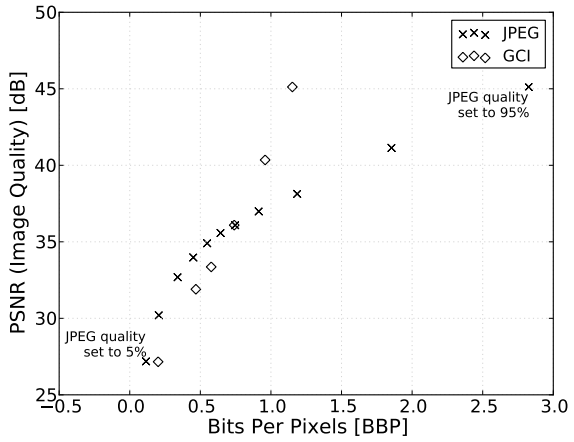
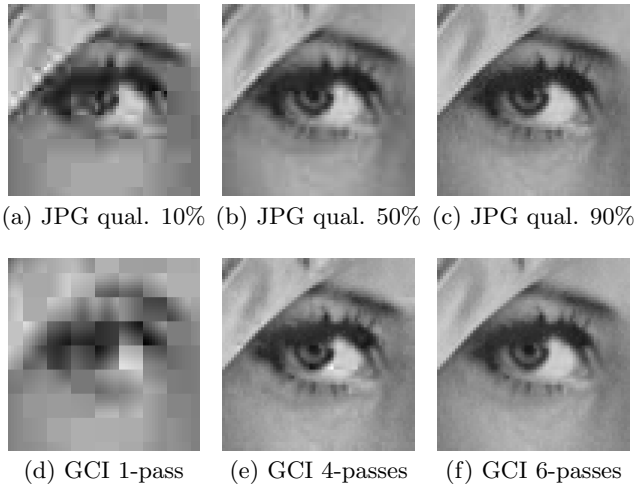


Figure 4: It is shown in the graph that GCI 6-pass encoding achieves a higher compression rate than JPEG above PSNR = 37. A higher PSNR value reflects a better image quality whereas a lower BBP value means higher compression. The BBP values for GCI shown include the memory required to store the 6 GRNs. In CGI, the ratio GRNs/tiles of the total amount of memory required is about 1/4.



(a) JPG qual. 10% (b) JPG qual. 50% (c) JPG qual. 90%

(d) GCI 1-pass (e) GCI 4-passes (f) GCI 6-passes

Figure 5: Comparison of JPG and GCI at low, medium and high quality settings.

algorithms, particularly in JPEG [5], against which the presented results are compared. Second, one pixel corresponds to one cell in the current implementation and it is easier to achieve smaller organisms that match smaller tiles.

5. GENE COMPRESSED IMAGE VS JPEG

In the first experiment it is investigated what quality and compression rates are achievable when compressing images with the proposed method (Algorithm 2). Bits per pixel (BPP) is a measure for image compression and has a value of 8 for 8bit grey scale, uncompressed images. In this paper the BBP value is simply calculated by dividing the resulting image file size by the total number of pixels. The quality of the compressed image is measured by calculating its peak signal-to-noise ratio (PSNR) to the uncompressed image [1]. Results are compared with JPEG in order to place them into the context of an established and optimised method.

The example image *Lena* is taken from the USC Signal and Image Processing Institute database [17]. The original version is a 512×512 pixel, uncompressed, 8bit grey scale TIFF image, hence, the image consists of 4096 8×8 pixel tiles. The maximum number of iterations of the ADS is set to 4096, which is the minimum when assuming each tile is different, since multi-pass encoding is used and in order to keep computation time low. Results for multi-pass encoded GCI images ($1 \ldots 6$ passes) are compared with JPEG images that are encoded with quality settings of $5, 10, 20, \ldots, 90, 95\%$ in Figure 4. As can be seen from Figure 4, applying multiple passes in GCI correspond to increasing the quality setting in JPEG. BBP and PSNR is calculated as described above for both GCI and JPEG images. Figure 5 shows example images for both methods using the lowest and the highest quality setting respectively. The file sizes of the compressed images are direct proportional to the BBP value. In the case of *Lena*, file sizes are: uncompressed TIFF $\sim$ 262 kByte, 95% quality JPEG $\sim$ 150 kByte and 6-pass GCI $\sim$ 37 kByte. The time required to optimise a GRN for one GCI pass is $\sim$ 10 hours on a 2.8 GHz Core2. 6-pass GCI encoding and decoding of an image takes about 1 second on the same hardware. It is found that the system reliably finds solutions when re-run, although this has not yet been quantitatively investigated.

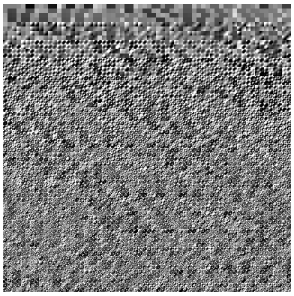
6. DYNAMICS OF THE GRN

In order to investigate the dynamic behaviour of the GRN, the output tile space of GCI pass 1 and 2 is visualised in Figure 6. As can be seen from Figures 6(a) and 6(b), only a relatively small part of the output tile space of pass 1 is used to represent the image. It is observed in Figure 6(b) that mostly tiles from early iterations of the developmental process are used. The reason for this is the fact that the current implementation creates tiles in the frequency domain. Therefore, the DC value of the tile and the lower frequency components are emphasised when calculating the fitness as those feature the largest coefficients. Once the brightness of all tiles in the image is matched by achieving basic tiles representing all 128 grey levels, the system is in a local optimum.

This behaviour is the main reason for using multi-pass encoding in order to increase image quality. In later encoding passes, the emphasis is more and more shifted towards higher frequency components due to the fact that only the differences to the tiles that are already produced in earlier passes need to be achieved. An example of this is shown in Figures 6(c) and 6(d). In this case, a larger part of the tile space is contributing to creating the image and the tiles used are not clustered at early developmental steps. However, using multi-pass encoding to increase image quality comes at the cost of increasing the file size of the GCI image, since for each pass, a new GRN, its parameters and the sequence of integers to encode the tiles needs to be stored.

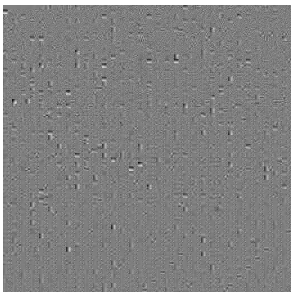
7. GENERALITY OF GCI

The results from Section 6 suggest that it should be possible to encode more than one image, particularly if they



(a) Tiles shown in the order in which they are developed in pass 1. Top-left tile is from iter. 1, top-right is from iter. 64, bottom-right is from iter. 4096.

(b) Image decoded using the iter. numbers saved at the encoding stage of pass 1 and a subset of the output tile space produced by the ADS (see Figure 6(a) on the left).



(c) Sequence of tiles shown in the order in which they are developed during pass 2.

(d) Image decoded using the iter. numbers from pass 2 and tiles from Figure 6(c)

Figure 6: Developed tiles (output tile space) and decoded image of pass 1 and 2 are shown. In passes 2–6 only the difference between the original image and the one already achieved needs to be developed in order to constantly increase the level of detail.

contain similar features, using the same set of GRNs. Theoretically, the output tile spaces of the 6 encoding passes used to encode *Lena* can be linearly combined to represent 4096^6 different 8×8 image tiles, which can be used to form a variety of different images. It is investigated whether it is possible to encode three images other than *Lena*, which are also taken from the USC Signal and Image Processing Institute database [17], namely *Elaine*, *Mandrill* and *Aerial01*. The same set of GRNs that is originally optimised for *Lena* is used to encode these images. Results of the GCI encoded versions of the three images are shown in Figure 8. The original size of the images is 512×512 pixels, hence, compression artefacts are only visible in the magnified versions that only show a part of the image.

The results show that it is possible to encode different images using the same set of GRNs. However, as can be seen from Figure 7, quality and compression rate of the GCI encoded versions of *Elaine*,*Mandrill* and *Aerial01* are lower than in the case of *Lena*, for which the set of GRNs is optimised. As one might expect, the *Elaine* image, which is most similar to *Lena*, achieves the best PSNR and BBP of the three. GCI encoding performs worse on the *Mandrill* and *Aerial01* due to greater level and characteristics of detail in those images.

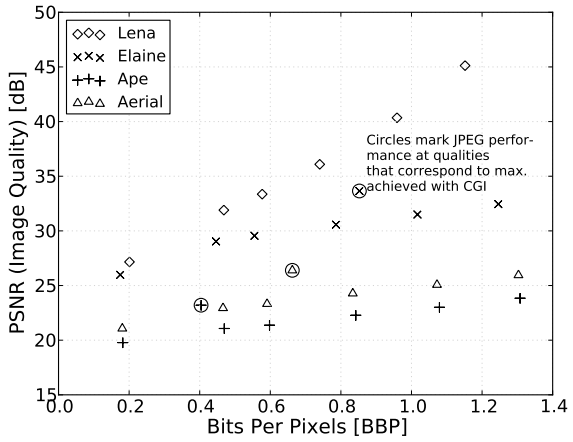


Figure 7: The same GRN, which is optimised for *Lena*, is used to encode different images, in order to investigate generality. PSNR achieved using multiple passes of GCI encoding is shown.

(d) Elaine (e) Mandrill (f) Aerial01

Figure 8: Different images encoded with the same set of GRNs are depicted. The bottom row shows details of the GCI encoded images from the top row.

8. DISCUSSION

A novel approach to image compression is proposed in this paper. It is shown that GRN based artificial developmental systems are suitable for image compression and outperform JPEG when optimised for a particular image. In the case of *Lena*, the compression rate is increased by a factor of 2.5 at the highest quality setting.

The behaviour of the ADS is investigated by visualising the output tile space of the developmental process. This provides insight into the way GCI works and reveals an important bias of the current implementation towards lower frequencies in early encoding passes. This bias is present due to the fact that the ADS is optimised to match tiles in frequency space, which puts more emphasis on the usually larger coefficients of the lower frequency components. This could be taken into account when improving the function of the structuring protein (Algorithm 1) in future experiments.

When working in frequency space, for instance, it might be beneficial to make this function dependent on the position of the cell within the 8 organism.

Furthermore it is shown that the set of GRNs, which is originally optimised for encoding *Lena*, is suitable to encode a variety of other natural images as well. As expected, the experiments undertaken show that the more features and characteristics similar to *Lena* are present in the image to encode, the better GCI compression performs both in terms of visual inspection and PSNR. When encoding 'unknown' images the higher compression rate than JPEG is generally not achieved, although if the goal is to encode a variety of different images, it will be beneficial to include a mixture of tiles from different images for the optimisation of the ADS.

There is also scope for future research on using ADSs for image compression. For instance, an interesting question is whether it is possible to reduce the number of encoding passes and thereby further increase the compression rate. A possible way of achieving this could be by parallelising multi-pass encoding by introducing multi-organism artificial developmental system.

One of the major drawbacks of the proposed approach at the moment is that it takes a lot of time and computational effort to optimise the GRN. However, there are scenarios where higher data compression is more critical than computational effort. Examples for such scenarios are remote applications, i.e. space applications where communication windows are physically limited, and robot applications where the amount of memory is limited. In the latter scenarios, the GRN could be optimised at the base station for images (or other sensor data) of the environment encountered by the field units, which would increase the data compression for transmission over time. Updated versions of the GRN can be transmitted at low cost, as they can be represented in a compact fashion and running the ADS is computationally cheap (the version in this paper already uses exclusively integer data structures and no multiplication or division), as opposed to optimising the system via an EA.

Another beneficiary of the image compression application presented in this paper could be the research field of artificial developmental systems, where a lot of work concentrates on pattern formation. The proposed algorithm could be suitable as a benchmark application for ADSs, which would be of more complex nature than achieving static patterns or shapes and would also put less constraints on the way the ADS explores the state space, since the order in which solutions are developed is not important.

9. REFERENCES

[1] W. Burger and M. J. Burge. *Principles of Digital Image Processing: Fundamental Techniques*. Springer Publishing Company, Incorporated, 2009.

[2] K. Clegg, S. Stepney, and T. Clarke. Using feedback to regulate gene expression in a developmental control architecture. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO) 2007*, pages 966–973, New York, NY, USA, 2007. ACM.

[3] A. Devert, N. Bredeche, and M. Schoenauer. Robust multi-cellular developmental design. In *Proc. of the 9th annual Conf. on Genetic and Evolutionary Computation (GECCO)*, pages 982–989, New York, NY, USA, 2007. ACM.

[4] N. Flann, J. Hu, M. Bansal, V. Patel, and G. Podgorski. Biological development of cell patterns: Characterizing the space of cell chemistry genetic regulatory networks. volume 3630 of *Lecture Notes in Computer Science*, pages 57–66, Canterbury, UK, September 2005. Springer.

[5] International Telecommunication Union. JPEG Standard (JPEG ISO/IEC 10918-1 ITU-T Recommendation T.81), 9 1992.

[6] J. F. Knabe, C. L. Nehaniv, and M. J. Schilstra. Regulation of Gene Regulation - Smooth Binding with Dynamic Affinity affects Evolvability. In *IEEE Congress on Evolutionary Computation (CEC 2008). Proc WCCI 2008*, pages 890–896. IEEE Press, 2008.

[7] S. Kumar and P. Bentley, editors. *On Growth, Form and Computers*. Elsevier Academic Press, 2003.

[8] T. Kuyucu, M. Trefzer, J. F. Miller, and A. M. Tyrrell. On the properties of artificial development and its use in evolvable hardware. In *IEEE Symposium Series on Computational Intelligence - IEEE SSCI 2009*, 2009.

[9] O. Leyser and S. Day. *Mechanisms in Plant Development*. Blackwell, 2003.

[10] H. Liao, Z. Ji, and Q. H. Wu. A novel genetic particle-pair optimizer for vector quantization in image coding. In *IEEE Cong. on Evolutionary Computation (CEC)*, pages 708–713, 2008.

[11] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *Communications, IEEE Transactions on*, 28(1):84–95, January 2003.

[12] A. Lindenmayer. Mathematical models for cellular interactions in development. i. filaments with one-sided inputs. *Journal of Theoretical Biology*, pages 280–299, 1968.

[13] H. Liu, J. F. Miller, and A. M. Tyrrell. Intrinsic evolvable hardware implementation of a robust biological development model for digital systems. In *Proc. of the NASA/DoD Conf. on Evolvable Hardware*, pages 87–92, Washington, DC, USA, 2005. IEEE Computer Society.

[14] J. F. Miller. Evolving developmental programs for adaptation, morphogenesis, and self-repair. In *7th European Conf. on Artificial Life*, pages 256–265. Springer LNAI, 2003.

[15] A. Momenai and S. Talebi. A fast evolutionary algorithm in codebook design. In *Proc. of the 4th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering Data Bases*, pages 1–6, Stevens Point, Wisconsin, USA, 2005.

[16] M. A. Trefzer, T. Kuyucu, J. F. Miller, and A. M. Tyrrell. A model for intrinsic artificial development featuring structural feedback and emergent growth. In *Proc. of the IEEE Congress on Evolutionary Computation (CEC)*, Norway, 2009.

[17] USC-SIPI. The usc-sipi image database.

[18] L. Wolpert, R. Beddington, T. Jessel, P. Lawrence, E. Meyerowitz, and J. Smith. *Principles of Development*. Oxford University Press Inc., New York, 2 edition, 2002.

[19] P. Yu and A. Venetsanopoulos. Improved hierarchical vector quantization for image compression. In *Circuits and Systems, 1993., Proceedings of the 36th Midwest Symposium on*, pages 92–95 vol.1, Aug 1993.