DOCTOR OF PHILOSOPHY

Study of the Fly Algorithm for 2-D and 3-D Image Reconstruction

Abbood, Zainab

Award date:
2017

Awarding institution:
Bangor University

PRIFYSGOL
BANGOR
UNIVERSITY

School of Computer Science
College of Physical & Applied Sciences

Study of the Fly Algorithm for 2-D and 3-D Image Reconstruction

Zainab Ali Abbood

Submitted in partial satisfaction of the requirements for the
Degree of Doctor of Philosophy
in Computer Science

Supervisor Dr. Franck P. Vidal

June 2017

Acknowledgements

First and foremost I would like to thank so much my supervisor **Dr. Franck Vidal** who made this thesis possible. I appreciate all his contributions of time, ideas to make my PhD experience productive and stimulating and for his corrections of this thesis. His positive outlook and confidence in my research inspired me and gave me confidence. He has taught me how can I grow as a researcher and be a good supervisor in the future.

I would also like to acknowledge the Ministry of Higher Education and Scientific Research (MOHESR) of Iraq for generosity in funding my PhD study.

Great thanks to Basra University for supporting and helping me during my undergraduate, master, and PhD study.

Many Thanks to my wonderful husband **Jassim Al-Autbi** his understanding and support during the research time. Without his encouragement, I would not be able to finish my PhD research.

A sweet thank to my kids **Ahmed** and **Amenah** to make my life during the study full with happiness and hope without despair.

I would like to many thanks my special and lovely dad who always his spirit surround and inspire me to pass the difficulties of the study.

I also would like to thank all my family in Iraq: beautiful mom, sisters, brothers, brothers in law, sisters in law, nephews and nieces to support my during my undergraduate, master, and PhD study. I am thankful to have such a lovely family.

Many thanks go to all the friends I met during these years, who made me and my family stay happy and enjoyable at Bangor. Last, but not least, I would like to thank all the staffs of School of Computer Science to help me a lot during these years and HPC Wales (`http://www.hpcwales.co.uk/`) for providing some of the computing facilities used in this study.

Statement of Originality

The work presented in this thesis/dissertation is entirely from the studies of the individual student, except where otherwise stated. Where derivations are presented and the origin of the work is either wholly or in part from other sources, then full reference is given to the original author. This work has not been presented previously for any degree, nor is it at present under consideration by any other degree awarding body.

Student:

Zainab Ali Abbood

Statement of Availability

I hereby acknowledge the availability of any part of this thesis/dissertation for viewing, photocopying or incorporation into future studies, providing that full reference is given to the origins of any information contained herein. I further give permission for a copy of this work to be deposited with the Bangor University Institutional Digital Repository, the British Library ETHOS system, and/or in any other repository authorised for use by Bangor University and where necessary have gained the required permissions for the use of third party material. I acknowledge that Bangor University may make the title and a summary of this thesis/dissertation freely available.

Student:

Zainab Ali Abbood

Abstract

The aim of this study is to investigate the behaviour and application of an evolutionary algorithm (EA) based on a particular approach of cooperative co-evolution algorithm (CCEA), the Parisian Approach. It evolves and keeps an entire population as an optimal solution to the problem instead of keeping only the best individual in classical EAs. The CCEA we selected is called the "Fly algorithm". It is named after flies, because the individuals are extremely primitive and correspond to three-dimensional (3-D) points. This algorithm has been relatively overlooked despite showing promising results in real-time robotic and image reconstruction in tomography. Our focus in this study is on two types of applications: medical imaging and digital art.

i) In the medical application, we aim to improve quantitative results in 3-D reconstructed volumes in positron emission tomography (PET). We investigate the use of density fields, based on Metaballs and on Gaussian functions respectively, to obtain a realistic output. We also investigate how to exploit individuals' fitness to modulate their individual footprint in the final reconstructed volume. An individual's fitness can be seen as a level of confidence in its 3-D position. The resulting volumes are compared with previous work in terms of normalised-cross correlation. In our test cases, data fidelity increases by more than 10% when density fields are used instead of using a naive approach. Our method also provides reconstructions comparable to those obtained using well-established techniques used in medicine (e.g., filtered back-projection (FBP) and ordered subset expectation-maximization (OSEM)).

Our algorithm relies heavily on the mutation operator. We propose 4 different fully adaptive mutation operators: basic mutation, adaptive mutation variance, dual mutation and directed mutation. Their impact on the algorithm efficiency is analysed and validated on PET reconstruction.

ii) In the digital art application, we present the first application of the Fly algorithm in digital art. This branch of digital art is called “evolutionary art”. The motivation is to evaluate the algorithm with a much more complex structure of flies. They are still defined as simplistic primitives (3-D points) but with colours, sizes and rotations. Different visual effects were investigated, such as mosaic-like images and spray paint rendering. An online survey (including 41 participates) was conducted to validate our approach. Participants compared our results with similar ones generated with open-source software (GIMP). Again, our method shows promising results.

In conclusion, our investigations confirm that the Fly algorithm works well with a complex search space. We demonstrate a fast and accurate solution to optimise a set of parameters in both applications. The Fly algorithm can improve reconstructed image quality compared to FBP and OSEM in medical application and to GIMP in digital art application.

Contents

List of Figures

List of Tables

List of Algorithms

Listings

Chapter 1

Introduction

1.1 Context

Computer vision (CV) is a "mature" discipline that is related to different subjects e.g. artificial intelligence (AI), signal processing, image processing, computer graphics (CG), etc. In the past few years, a number of academic researches have worked out in CV, with a focus on pattern recognition for autonomous vehicles. It is beginning to become practical and successful. Since its earliest days, CV has shown progress in traditional applications, such as robotics and medical imaging. Today, CV keeps growing to produce non-traditional applications such as scientific analysis, art and many more [171, 147]. In general, CV relates to processing, analysing, reconstructing and understanding digital images. In this thesis, we focus on the applications of CV that are related to image-based, three-dimensional (3-D) reconstructions in medicine (see Chapter 5) and two-dimensional (2-D) reconstructions in digital arts, which can be found in Chapter 7. The image reconstruction topic has been well studied and the earliest example was reported in the earliest days of machine vision. In recent years, due to dramatic improvements in computational power and the increased ability of digital imaging technology, the reconstruction of shapes has received interest as a practical application. Image based reconstruction is an ill-posed inverse problem. This problem can be solved as an optimisation problem [156]. In this thesis, we use evolutionary algorithm (EA), in particular the Fly algorithm [102], which offers a solution to our reconstruction problems. As a result, the Fly algorithm works on complex problems, directly takes into account the imaging geometry and can be considered as an intrinsic parallelism algorithm. However, the big challenge is to find methods to tune EA parameters to work with complex data. This will be one of the focuses of this thesis.

1.2 Hypothesis

Evolutionary computation is a branch of research in mathematical optimisation. It is useful for solving complex (ill-posed) problems that have ambiguity in the shape of search space (e.g. it is unknown or extremely irregular) as well as when you do not want (or cannot make) too many unreasonable assumptions. Because of the stochastic nature of EAs, they can adapt to various problems. However, there might be difficulty to express the optimisation problem in an evolutionary framework in terms of defining a suitable individual encoding, selection method, genetic operators, and fitness function. Choosing an appropriate EA technique may be a challenge in imaging and CV, and it is a growing research topic in itself. Most people are familiar with simple genetic algorithms (binary encoding of individuals) and real-valued genetic algorithms used as black-box optimisation tools. Consider an optimisation problem that consists in finding the best possible 3-D position of N points. The search space has $3N$-dimensions. When a black-box genetic algorithm is considered, there will be k individuals in the population, with N genes per individual. When is N is large, this approach will fail due to the computing time that will be required. A more recent class of algorithms is cooperative co-evolution algorithm (CCEA). Using this framework, it may be possible to only require N individuals, with 3 genes per individual. The hypothesis of this thesis tries to demonstrate it and open the door for a rather new EA algorithm for the scientific communities.

> *A relatively overlooked algorithm (Fly algorithm) can provide a competitive alternative (e.g. in term of accuracy, ease of use [1]) to some of the most traditional approaches used in image (2-D or 3-D) reconstruction when considered as an inverse problem.*

This thesis is an investigation of this hypothesis in the context of tomography reconstruction in nuclear medicine and digital arts.

[1] One of the main problems with the reconstruction methods based on maximum-likelihood expectation-maximization is the lack of clear stopping criteria (see Figures 5.25 and 5.21 for an illustration).

1.3 Objectives

To investigate this hypothesis, the main aim of this thesis will be to examine the impact of the Fly algorithm in order to produce a high quality image reconstruction in both medical and art applications. Therefore, the objectives of the study are as follows:

1. The study tries to improve the Fly algorithm for positron emission tomography (PET) in [156] specifically in the voxelisation stage.

2. To examine the Fly algorithm results using different mutation operators.

3. To investigate the use of the Fly algorithm in a new application dedicated to artistic rendering.

4. To examine different structures of flies from simple to complex structures.

5. To compare whether the Fly application performs better than some traditional methods.

1.4 List of publications

Below is a list of published articles that present some parts of the work.

1. **Z. Ali Abbood**, J.-M. Rocchisani and F. P. Vidal, *Visualisation of PET data in the Fly algorithm*, in Eurographics Workshop on Visual Computing for Biology and Medicine, 2015, pp. 211–212.

2. **Z. A. Abbood**, O. Amlal and F. P. Vidal, *Evolutionary art using the Fly algorithm*, in Applications of Evolutionary Computation: 20^{th} European Conference, EvoApplications 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings, Part I, G. Squillero and K. Sim, Eds. Cham: Springer International Publishing, 2017, pp. 455–470, ISBN: 978-3-319-55849-3. DOI: 10.1007/978-3-319-55849-3_30.

3. **Z. Ali Abbood**, J. Lavauzelle, É. Lutton, J.-M. Rocchisani, J. Louchet and F. P. Vidal, *Voxelisation in the 3-D Fly algorithm for PET*, Swarm and Evolutionary Computation, 2017, issn: 2210-6502. doi:10.1016/j.swevo.2017.04.001. In Press.

4. **Z. A. Abbood** and F. P. Vidal, *Basic, Dual, Adaptive, and Directed Mutation Operators in the Fly Algorithm*, Proceedings of Artificial Evolution, 2017. In Press.

5. **Z. A. Abbood** and F. P. Vidal, *Fly4Arts: Evolutionary Digital Art with the Fly Algorithm*. Art and Science. Submitted.

In addition, three pictures (see Figure 1.1) have been submitted to *Art & Science in Evolutionary Computation*, a join side event for EA2017 conference with Galerie Louchard in Paris.

Figure 1.1: Pictures submitted to 'Art & Science in Evolutionary Computation' organised by the EA2017 conference and Galerie Louchard, Paris.

1.5 Contributions

The main contribution of this thesis is to reduce the computing cost and produce 2-D/3-D reconstructed datasets while retaining the required quality. The methods are based on the co-evolution strategy (also called the Fly algorithm). It will focus on the major contribution of this work:

Extraction of the solution in the Fly algorithm for PET . Over the last decade, PET has had a potential impact in cancer management. However, there are still

limitations to using traditional reconstruction methods, such as filtered back-projection (FBP) and ordered subset expectation-maximization (OSEM) (with different levels of noise and poor resolutions). We develop a 3-D reconstruction method to solve the problem as an optimisation problem. Our proposed method shows promising results when compared with traditional methods (see Section 5.5). This work can be found in publication no. 3.

Improving medical imagery accuracy in voxelisation . We investigate the use of density fields to obtain more realistic outputs for PET data reconstructed using the Fly algorithm. It is CG techniques that we adapt to limit the noise in PET volume data (see Section 5.4.1). This work can be found in publication no. 1 and 3.

Different EA mutation operators . We propose four different fully adaptive mutation operators: basic mutation, adaptive mutation variance, dual mutation, and directed mutation. We address their impact on the algorithm's efficiency, which is analysed and validated on PET reconstruction. Our algorithm is a self-adaptation algorithm that controls the setting of the EA parameters themselves, embedding them into an individual's structure (genome) and evolving them. This work can be found in publication no. 4.

Digital art model . We propose a technique to generate artistic images. This method draws from CG, EA (in particular, the Parisian evolution approach and the Fly algorithm) and scientific computing. Our implementation produces several visual artistic effects by taking advantage of using the Open Graphics Library Shading Language (GLSL) to avoid the limitation of the fixed graphics rendering pipeline and to benefit from programmable graphics hardware. It is possible to update the content of the 2-D texture in real time to change the texture of the flies. We use the high-level programming language GLSL to take control over critical stages of the graphics rendering pipeline. This work can be found in publication no. 2 and 5.

1.6 Outline

This thesis investigates the EAs, particularly the Parisian evolution approach (Fly algorithm). This thesis is organised into eight chapters. The first chapter gives information of this research, including motivations; it highlights the main contributions of this study and gives a list of published papers. Chapter 2 presents the scientific context related to this work and the main applications that have been used in this thesis. Chapter 3 provides a brief overview of EAs. A review of the previous work focusing on EAs and their applications to image processing is also presented. The chapter ends by introducing the challenges of working with EAs. The following chapter presents a special type of CCEA, which is the Parisian approach (specifically the Fly algorithm). The most common applications based on the Fly algorithm are indicated. The implementation of the Fly algorithm to reconstruct a 3-D image of PET scanner is shown in Chapter 5. We investigate the use of density fields for the first time with the Fly algorithm, based on metaballs and on Gaussian functions respectively, to obtain a realistic visualisation output. The results of our algorithm show that the performance of using density fields is competitive compared with those of FBP and OSEM, which are traditionally used in nuclear medicine. Chapter 6 proposes four different fully adaptive mutation operators: Basic Mutation, Adaptive Mutation Variance, Dual Mutation and Directed Mutation, and it addresses their impacts on the algorithm's efficiency. Chapter 7 produces a new implementation, digital arts, working for the first time with the Fly algorithm. This chapter adds different ideas that have been used with previous applications of the Fly algorithm. Our technique inherited from CG, artificial evolution, and scientific computing. The last chapter discusses the work carried out and provides some possible directions for future work. Further, Appendix A gives details of the programming frameworks implemented on graphic processing unit (GPU). The appendix focuses on the most common GPU programming language used in this thesis (see Section A.3). Shader programs used in Chapter 7 are presented in Appendix B.

Chapter 2

Background: Scientific Context

2.1 Introduction

Digital image processing has been widely used in different scientific areas, including medicine, aerospace, and chemistry, to name a few. Although it has been established for a long time, it is still a fast-growing topic that takes advantage of new developments in computer processors and mass storage devices. Digital image processing can greatly benefit more traditional fields, such as medicine, video production, photography, painting, etc. by also taking advantage of new developments in CV, AI, and machine learning (ML). As a result, computing gives new opportunities to examine and process massive amounts of digital image data every day [25, 96]. Computer-aided diagnosis (CAD) is a very active research topic, with well-established annual conferences, including Medical Image Understanding and Analysis (MIUA). More recently, the Eurographics Association launched an annual Workshop on Intelligent Cinematography and Editing (WICED) [1].

Image processing is a broad scientific area that includes subfields such as image enhancement, feature detection, image segmentation, image classification, image registration, and 2-D or 3-D image reconstruction [4]. All of these fields include complex methods that may fail due to disturbances in the data such as noise. In this thesis, we will focus on image reconstruction. It is often an inverse (ill-posed) problem: the solution of inverse problem if it exists may not be stable, or unique. Inverse problems can be tackled using optimisation algorithms such as artificial evolution (AE) [156], simulated annealing (SA) [97], local search algorithms, etc. This chapter presents the general application context related to this thesis, beginning with a brief introduction

on image processing and CV followed by a definition of the main applications, which include tomographic reconstruction in nuclear medicine and mosaic art.

2.2 Imaging

Digital imaging can be described as the application of some operations to process digital images in order to enhance or extract useful information from them. Ideally, this is done by computers, with little or no human intervention [59].

Most images are the result of a measurement of some physical phenomena, such as light, heat, distance, or energy. Measurements can be recorded in any digital format. A digital image is considered a 2-D discrete-space signal. An image is made of a limited number of cells, which are usually called picture elements (pixels). Each pixel has a spatial position and at least one numerical quantity (e.g., an intensity) associated to it. The most classical representation of a digital image is a 2-D grid ($f(x, y)$), in which an image's pixels are identified by a pair of coordinates (x, y) and a value. This value often corresponds to the light intensity (or brightness), that is, the light density of a tiny region of the image. Figure 2.1 models the light density to pixel value conversion [59, 67].

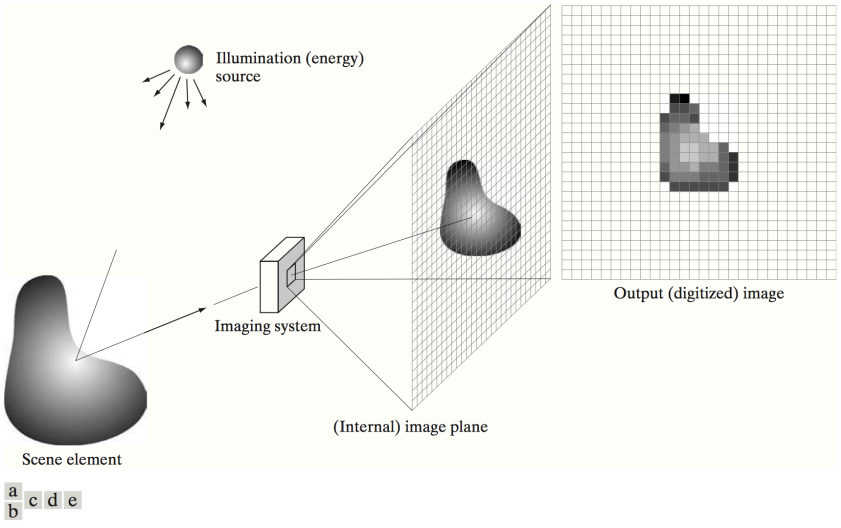


Figure 2.1: An example of the digital image processing of grey-scale image: (a) energy source, (b) item of an image, (c) imaging system, (d) projection of the scene onto the image plane and (e) digitised image(Image from [67], courtesy of Gonzalez).

We considered the simplest model in Figure 2.1: digital greyscale images. However, a colour image quantifies the light components and assigns a colour to each pixel. In this case, a pixel is a vector of a mix of colour elements. There are different models to represent colour images, including the red-green-blue (RGB), the cyan-magenta-yellow (CMY), the cyan-magenta-yellow-black (CMYK) and hue, saturation, intensity (HSI) colour models. RGB is a colour space that maps the emittance of light. RGB is an additive colour spectrum, and when all the primary colours are combined, it forms white, which is the emission of all primary colours. Most electronic displays (computers, phones, media players, televisions, etc.) are RGB models, since they use the emittance of light to create colours (i.e., the pixels have little sub-pixels that show only red, green or blue). However, devices like printers and copiers do not emit light. They absorb it and print colour using the CMYK model (the colour spectrum known as subtractive colours). When all the primary colours are combined, they form black, which is the absorption of all light [67].

After an image has been acquired, either in greyscale or colour, digital image processing can be used to enhance and/or analyse it using a computer. In their famous book [67], Gonzalez and Woods classify three levels of the imaging process: low, mid and high levels. The first corresponds to filtering. It takes an image as an input and returns an image as an output. It is mainly used for image enhancement, noise removal, restoration and compression. However, mid-level processing focuses on the extraction of information for different implementations, including image segmentation, registration, matching and classification (recognition). Obviously, these applications are characterised by images in the input phase, but the outputs are features that are extracted from these images. Finally, high-level processing is associated with extracting semantic meaning from images. It relies on advanced techniques using AI and ML. The input is an image or a set of images, and the output is the analysis of the input. Object detection, recognition, shape analysis and tracking [67, 30] are the aims of CV.

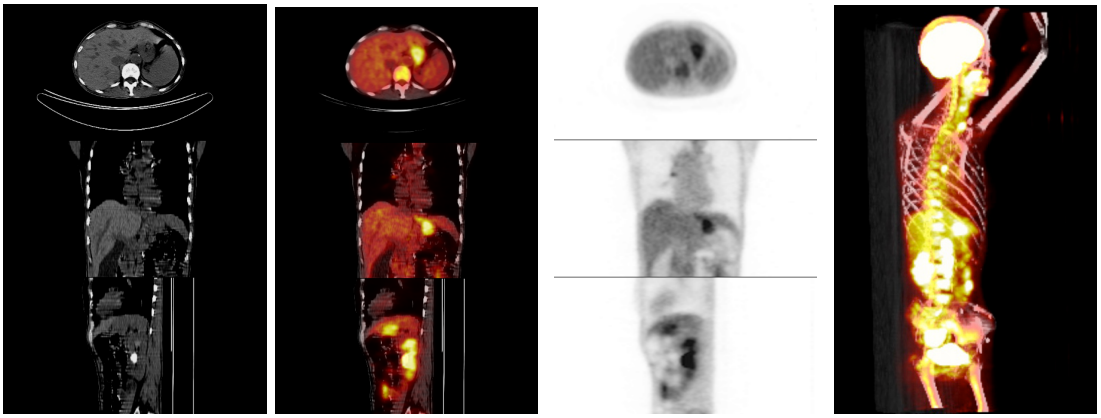
2.3 Computer vision

The aim of CV is to develop a computerised system that is able to analyse and understand the real world by processing one or several 2-D images. In other words, it is the science of building a computer system that has humanlike perception [165, 164]. Typical CV applications include motion estimation, 2-D or 3-D reconstruction, object segmentation, etc. They provide the computer with natural skills, such as self-improvement, learning, reliability and controlling results to gain optimal solutions. The ultimate goal is to allow the computer to make a decision without any human intervention. From the point of view of computing, it is important to mimic high-level intelligence and skills algorithmically. Some CV applications rely on mathematical optimisation to solve complex and ill-posed inverse problems using global optimisation methods. Those inspired by evolutionary processes observed in nature (such as reproduction, mutation, recombination and selection) are not unusual in CV [30]. A CV system requires a high rate of processing and may rely on a large database of information in terms of digital images. In addition, challenges arise in handling the noise in the image signal itself [151, 165]. Thus, from a software development point of view, dealing with CV requires a combination of several sophisticated techniques, such as object-oriented programming, image processing, parallelism, ML (also known as pattern recognition), AI, etc.

From the above definitions, CV heavily relies on image processing, in particular, the high-level stage of image processing [151]. The next chapter surveys the application of image processing in each processing level as well as that of CV, in which EAs are given.

2.4 3-D Tomographic reconstruction in nuclear medicine

The first application of our project is 3-D tomographic reconstruction in nuclear medicine. Various tomography techniques exist in medicine, most of which have modalities using radiation. This include computed tomography (CT), cone-beam computed tomography (CBCT), single-photon emission computed tomography (SPECT) and PET. CT is



(a) CT **(b)** PET and CT **(c)** PET **(d)** Visualisation

Figure 2.2: PET-CT examination of a cancer patient: CT provides anatomical information, while PET provides physiological information (data available on The Cancer Imaging Archive (TCIA) at `https://public.cancerimagingarchive.net/`).

intensively used in radiology departments to examine any part of the body. CBCT is a more recent 3-D technique that provides a high spatial resolution this is useful for head examinations in dentistry or radiotherapy. SPECT and PET are specifically used in nuclear medicine departments (see Fig. 2.2).

CT and CBCT are forms of transmission tomography, in which the source of radiation is located outside the patient. The reconstruction provides high resolution anatomical data. SPECT and PET are forms of emission tomography (ET), in which the source is made of radioactive molecules injected in or inhaled/ingested by the patient. Both, therefore, occur within the patient. The reconstruction aims to provide an estimation of the radioactive distribution within a patient in relation to the uptake of these radioactive molecules depending on a physiological process (e.g., tumour gross or bone fracture). The images reconstructed by SPECT and PET have a much lower resolution and poorer signal-to-noise ratio (SNR) than CT or CBCT.

X-ray photons are used in both CT and CBCT. γ-rays are used in SPECT, and positrons (β^+ or e^+) are used in PET. When a β^+ particle combines with an electron (e^- or β^-), it may result in an annihilation reaction, producing two photons of 511 keV in opposite directions. Detected pairs of photons are used during the reconstruction process in PET.

Tomography is a multi-angular data acquisition process followed by mathematical reconstruction (see Fig. 2.3). The data acquisition consists of obtaining many projections from different angles. The image produced by the concatenation of the successive

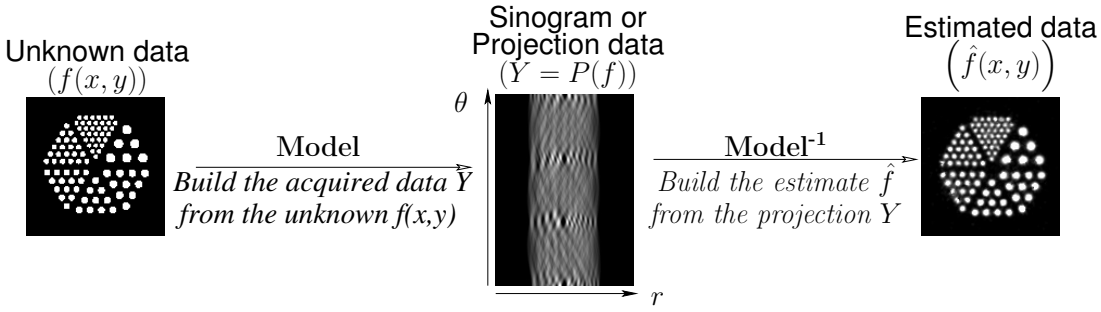


Figure 2.3: Principle of tomography .

projections is usually called a sinogram (see the middle image in Fig. 2.3). The projections correspond to integral quantities along straight lines from the source to the detector. The reconstruction process is a mathematical routine that consists of back-projecting the measurement data into the object's space. This process is an inverse problem, with ill-conditioned data due to the photonic noise (Poisson noise) affecting the data. Noise is a major concern in ET as well as in CT in low-dose conditions conducted to minimise patient irradiation. The computational speed of image reconstruction is another factor to be studied in order to improve the throughput of examinations. Some drawbacks in image quality need to be corrected, such as streak artefacts in CT [153] or the scattering of photons. Although effective and fast methods have been developed to solve 2-D reconstruction problems (that is, to compute a single planar slice), solving 3-D reconstruction problems (as in CBCT or C-arm tomography) is still challenging and computationally expensive. A practical way to solve 3-D reconstruction problems is to consider them as set of 2-D problems.

It is usual to describe two classes of reconstruction algorithms: 1) analytic reconstruction methods, and 2) iterative reconstruction methods (including algebraic and statistical based methods).

Analytic reconstruction methods are used in CT, while statistical reconstruction methods are used in ET. Analytic reconstruction methods are based on continuous modelling. Reconstruction consists of the inversion of measurement equations (such as the Radon transform). The most frequently used is the FBP algorithm. The sinogram, which is the observed measure, is built from a set of planar projections at successive angles. Tomographic reconstruction consists of estimating the original object from this sinogram. Analytic methods inverse the Radon transform using the Fourier slice theorem, in which the 1D Fourier transform of a projection is equal to a slice of the 2-D Fourier transform

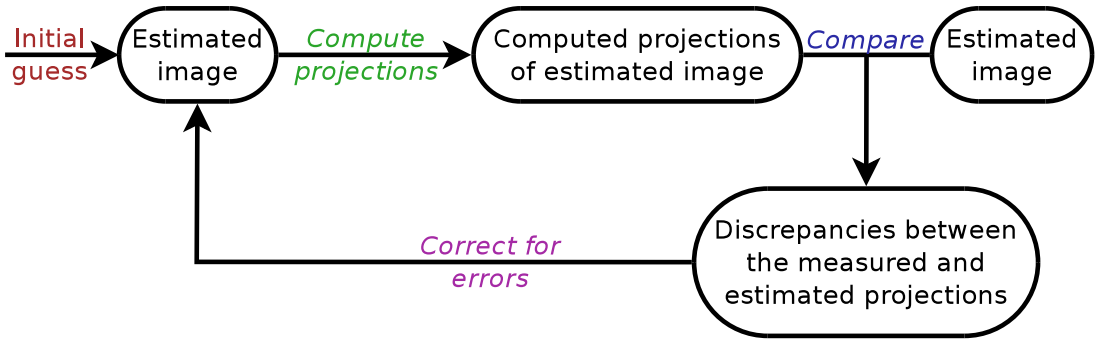


Figure 2.4: Steps in iterative algorithms.

of the original image. The complete 2-D Fourier transform of the image is reconstructed from these 1D Fourier transforms. Then, the image is obtained by inverting its Fourier transform.

Iterative reconstruction techniques are based on iterative correction algorithms. They follow a general scheme that consists of estimating a new image at a certain step by combining 1) the image estimated at the previous step and 2) the data generated from this estimate (see Fig. 2.4). This process is repeated until a given criterion is satisfied. The algebraic reconstruction technique (ART) was one of the most commonly used iterative algorithms at the start of transmission tomography [68] and is again used in combination with some statistical models thanks to the power of recent computers. It consists of updating the reconstructed volume for each measurement pixel location ray by ray. There are many other iterative techniques, including the simultaneous algebraic reconstruction technique (SART), the multiple algebraic reconstruction technique (MART), simultaneous iterative reconstruction technique (SIRT). For more information about tomographic reconstruction, refer to the literature [10, 13, 63]. Note that, in PET, raw list-mode data can be rebinned into sinograms to take advantage of conventional reconstruction algorithms [53], although better images can be reconstructed using dedicated codes [85, 100, 148].

In SPECT and PET, maximum-likelihood expectation-maximization (MLEM) [140] and its derivatives are now more popular than the previous methods [131]. The main reason is that they take into account Poisson noise in the measured photon count. OSEM has become the standard reconstruction method in PET for this reason [81]. Its principle is to reduce the amount of projections used at each iteration of the EM algorithm by subdividing the projections into K sub-groups. The projections of a sub-group are

uniformly distributed around the volume to reconstruct. One of the main problems of MLEM and its derivatives, including OSEM, is the difficulty of choosing a good stopping criterion [21]. MLEM initially converges towards an acceptable estimation of tracer distribution. Then, when the number of iterations increases, the reconstruction becomes noisy.

In this thesis, we are interested in ET in nuclear medicine because of its poor spatial resolution and relatively high noise, making reconstruction an ill-posed inverse problem. We propose to solve it as an optimisation problem using evolutionary computing. In particular, we focus on tomography reconstruction in PET, which is taking over SPECT in clinical practice.

2.5 Mosaics-like and Painterly Rendering

Medicine is not the only field that benefits from technological advances in digital imaging and computer vision. For example, the boundaries between artists and computer scientists may become thinner as technology becomes more and more ubiquitous. A relatively new field of CG is non-photorealistic rendering (NPR). One of the main goals of NPR is to produce "digital art" that can benefit the artistic community as well as the scientific community, for example, in scientific and medical visualisation [83]. Rendering algorithms have been proposed to simulate multiple forms of traditional art, including digital watercolours [45], line art [98], expressive painting [39] and Celtic art [86]. This thesis focuses on the most ancient of classical art forms, mosaics, and painterly rendering. This thesis also includes other arts forms like spray graffiti.

A digital mosaic aims to provide artistic touches to a source image by covering it by tens, hundreds, or thousands of small-coloured square tiles in a way that resembles ancient mosaics or stained-glass windows. To design a mosaic, an artist needs to precisely decompose the original image into tiles with different sizes, colours and orientations. The artist requires a large area to fit the tiles together like a jigsaw in order to form an image (i.e., it is not unusual for mosaics to encompass over several square metres) [51,

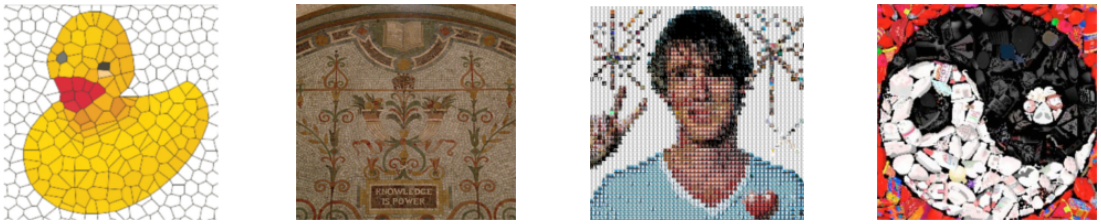


Figure 2.5: Mosaic classification: (a) crystallisation mosaic, (b) ancient mosaic, (c) photo mosaic and (d) puzzle image mosaic. (a) courtesy of DoBashi [47], (b &c) from the open images dataset (`https://github.com/openimages/dataset`) under the CC BY 4.0 license, and (d) courtesy of Gallo [22].

16]. The main goal of digital mosaic is automatically to generate a discrete coloured image that still gives the same impression as the real image.

In image processing and computer vision, the approach consists of building an algorithm that automatically produces an image with mosaic effects that has as little user intervention as possible. The produced mosaic image should replicate the features of the real image [56]. One of the difficulties in digital mosaic generation is that the original image may be visualised into various mosaics. Therefore, choosing the appropriate tile data set (including tile number, position, size, colour and rotation) allows for an appropriate final mosaic.

Figure 2.5 shows that mosaic images can be categorised into one of four types: 1) a crystallisation mosaic, 2) an ancient mosaic, 3) a photo mosaic or 4) a puzzle image mosaic. The first two types are traditional reconstructions of real images using small tiles. Although traditional mosaics use square tiles, in digital mosaics more complex shapes can be used. The last two are obtained by decomposing an original image using multiple small images.

To our knowledge Haeberli is the first researcher who worked on digital mosaic [69]. He created attractive images using an ordered collection of brush strokes to create mosaic and paint effects. He generated images by regulating the colour, shape, size, and orientation of individual brush strokes. To control the mosaic effect his method heavily relies on Voronoi diagrams. One of the main limitations of his algorithm at the time is that it took several hours to produce a satisfactory image. However, much less time should be required with today's "massively parallel processors". This is actually the approach followed by Hoff and his colleagues to overcome the limitation mentioned

above. They presented an implementation to compute discrete Voronoi diagrams on GPUs [76]. The method starts with a set of random points representing various sites in the image. They are used as the basis to create polygonal meshes that can be rendered in OpenGL to create the Voronoi diagrams. Their approach relies on a metrics based on the Euclidean distance for each site, which computes the distance from any point to that site. Each site has a unique colour.

Hausner improved Hoff's method to use regular and square tiles only. The aim is to create images that have an effect similar to actual mosaics [74]. Each tile may have a different size, colour, and orientation based on the image considered. This approach relies on Centroidal Voronoi diagrams, which usually order points in regular hexagonal grids. Instead of using the Euclidean distance as a metrics, the Manhattan distance is preferred to place the tiles in different orientation following the edges of the original image.

Lai *et al.* [93] extended Hausner's work by trying to place mosaic tiles on a surface. The tiles are located over a mesh model that is created using a Centroidal Voronoi diagram and the Manhattan distance. The size of tiles is regular, i.e all the tiles have the same shape (rectangle) and size. The orientation of tiles depends on a vector field, which is interpolated over the surface based on control vectors. The algorithm is sensitive to sharp creases, open boundaries, and boundaries between regions of different colours, which may affect the orientation of tiles.

Lu *et al.* presented a hybrid method that combines Centroidal Voronoi Tessellation (CVT) and Monte Carlo with minimisation (MCM). CVT places the tiles on a mesh surface. Because of local minima, MCM is applied to optimise the result of CVT on a global basis, which improves the final results [107].

In 2015, Hu and his colleagues presented an algorithm for the reconstruction of digital surface mosaics based on irregularly shaped tiles [80]. They use a hybrid optimisation paradigm, which includes continuous configuration optimisation and discrete combinatorial optimisation. In the continuous configuration optimisation scheme, the tiles are adjusted using iterative relaxation. The aim is to adapt their position, orientation, and scale to fit onto approximated Voronoi regions. The aims of

the discrete combinatorial optimisation are to reduce the amount of overlapping tiles and to increase the surface coverage.

Nguyen *et al.* [116] produced digital images using an EA based on multi-dimensional archive of phenotypic elites (MAP-Elites). Their aim was to demonstrate that deep neural networks (DNNs) can be easily fooled. Their implementation evolves a population to produce a tremendous diversity of images with a strong chance that DNN can classify the objects correctly.

Kim and Pellacini [90] presented a new kind of mosaic that they called jigsaw image mosaics (JIM). The aim of JIM is to create an output image from an input image and images of small objects (e.g. sweets and shells). The images of small objects are the tiles. Each tile has it own polygonal shape and pattern. The authors rely on a general energy-based framework, which extends existing algorithms such as Photomosaics and Simulated Decorative Mosaics, to minimise the error between the reference image and JIM. Centroidal Voronoi diagramss (CVds) are used to maximise the alignment of the shape borders.

An extension of this research topic is called "Painterly Rendering". Painterly styles are different from photographs and photorealistic rendering. In Painterly Rendering an algorithm places brush strokes on specific image regions and provides the perception of depth by changing the colour, size, shape and position of strokes [75].

In 2009, Zeng and his colleagues presented a stroke-based painterly rendering algorithm. The algorithm places the strokes depending of the segmentation of the reference image using recent image parsing techniques from CV. These techniques recognise object categories (e.g. people, clothes, sky, trees, etc.) and decompose an original image into a hierarchy of segmentation components in a parse tree representation. Then, the brushes are placed on the canvas guided by the image semantics included in the parse tree. The colour of stroke is recovered from the reference image [173].

Another approach in painterly rendering is the use of Evolutionary art [42]. In this context, an EA somehow generates images by placing brush strokes. Artificial evolution is used to modify the size and position of brush strokes by mutation and recombination. The colour of stroke is recovered from the reference image. Evolutionary art is often an

interactive task where the user/artist plays the role of a selection operator. Our work follows the Evolutionary art paradigm. We provide a method without the need of any user interaction, without constraints such as the requirement to generate a Voronoi diagram, and limit the amount of *a priori* knowledge to the input image.

In this thesis, we revisit digital mosaic-like image generation. Our method is suitable for several stylistic effects such as spray paint. Image generation is considered an optimisation problem; we propose to solve it using AE, in particular, CCEA. Our method relies on the Fly algorithm [102].

Chapter 3

Background: Evolutionary Approaches

3.1 Introduction

Artificial evolution describes a wide class of unbiased optimisation and search methods. It has been initially developed as a general purpose optimisation method. It is used to explore a search space, whatever it may look like, and find the best possible solution to the corresponding optimisation problem. Artificial evolution relies on Darwin's principles to mimic complex natural behaviours, which are stochastic in nature [14]. AEs have many advantages over traditional methods in image processing and computer vision applications: they require less domain-specific information, are easy to use, work on complex problems, directly take into account the imaging system geometry, and can be easily implemented to take advantage of parallelism on a set of solutions due to the intrinsic nature of the algorithm. Also, they may need less human interaction.

EAs have been utilised as adaptive meta-heuristic random search methods to find an optimal solution toward final solution [33]. They can be problem-independent and implemented as black-box optimisation tools. These algorithms are used in both homogeneous and heterogeneous systems, e.g. with fixed or variable numbers of genes on each individual respectively [71]. Equally important, it is a convenient technique in problems with a large search space and in unreliable environments (e.g. with noisy and incomplete data). EAs may work when some classical (deterministic) algorithms fail (e.g. gradient descent). In [157], an EA is used to tune a complex mathematical model of respiration and its results are compared with those obtained with downhill simplex

and conjugate gradient descent methods. As mentioned previously, artificial evolution can be used as black-box optimisation; however, most successful EAs depend strongly on parameter settings [64, 31, 170, 30], which is frequently problem-dependent.

In Figure 3.1, we use Google Scholar to assess the number of publications in *"image processiong" OR "computer vision"* and how many of them use evolutionary computing. Every bar in the chart represents a two-year period. The last time period considered is the last possible one to-date, i.e. 2014-2016. It can be seen that a rapid growth occur from the early 70s to the early 90s in imaging, then a plateau. Over the past two decades, AE has played a growing role in imaging and computer vision, thanks to the advantages mentioned above and to technological advances in computer hardware. AE can be classified as a general-purpose optimisation method and has been applied in a wide area of scientific fields. A good illustration of this is the European Conference on the Applications of Evolutionary Computations (EVOApplications), which covers a variety of scientific applications where EAs can be used. EVOApplications introduces several tracks, each concentrating on an area of application of genetic and evolutionary computation and other related computational intelligence fields. These include: Natural Computing Methods in Business Analytics and Finance (EvoBAFIN), Evolutionary Computation, Machine Learning and Data Mining in Computational Biology (EvoBIO), Nature-inspired Techniques for Communication Networks and other Parallel and Distributed Systems (EvoCOMNET), Evolutionary Algorithms and Complex Systems (EvoCOMPLEX), Evolutionary Algorithms in Energy Applications (EvoENERGY), Bio-inspired Algorithms in Games (EvoGAMES), Evolutionary Computation in Image Analysis, Signal Processing and Pattern Recognition (EvoIASP), Evolutionary and Bio-Inspired Computational Techniques within Real-World Industrial and Commercial Environments (EvoINDUSTRY), Knowledge Incorporation in Evolutionary Computation (EvoKNOW), Bio-inspired algorithms for continuous parameter optimisation (EvoNUM), Parallel Architectures and Distributed Infrastructures (EvoPAR), Nature-inspired algorithms in Software Engineering and Testing (EvoSET), Evolutionary Robotics (EvoROBOT), Evolutionary Algorithms in Stochastic and Dynamic Environments (EvoSTOC), Genetic Programming (EuroGP), Evolutionary Computation in Combinatorial Optimisation (EvoCOP), Computational Intelligence in Music, Sound, Art and Design (EvoMUSART). The longest running of all EVOApplications tracks is EvoIASP, which began in 1999 [11]. Our paper on

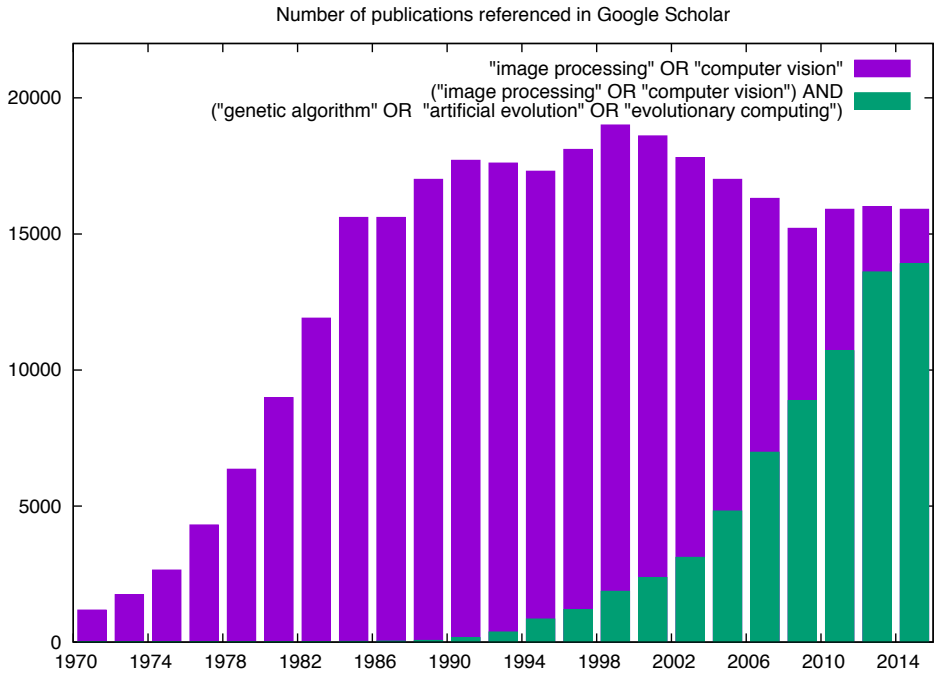


Figure 3.1: Growth of the use of artificial evolution in imaging and computer vision.

"Evolutionary Art using the Fly Algorithm" has been accepted for the 17^{th}EvoIASP although it was also suited for EvoMUSART (see Chapter 7).

This chapter includes a comprehensive reference list and summarises the important work and latest achievements in the AE field and their application in imaging and computer vision. It is organised as follows: Section 3.2 reviews in detail how EAs can be built. Section 3.3 focuses on AE applied to image processing applications such as image segmentation, image enhancement, image stereo matching, and image reconstruction. The chapter ends with a conclusion and highlights the current and future challenges in this domain.

3.2 Evolutionary algorithms

Evolutionary algorithms have been well studied, both empirically and analytically, and it has been demonstrated that there are different stochastic optimisation methods based on Evolutionary algorithms depending on how the EA represents the problem to solve.

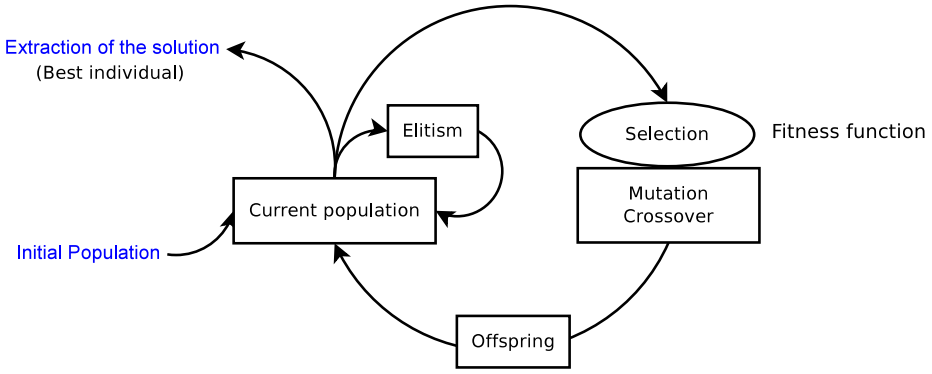


Figure 3.2: General layout of EA

This section introduces the concepts of EA needed for the understanding of the CCEA on which this thesis relies, namely the Fly algorithm initially proposed by Dr. Jean Louchet in 1999 [104]. More details on the Fly algorithm are available in Chapter 4.

The basic rules of EA were first introduced by Holland at the University of Michigan in the 1970s [92, 17, 24]. However, the origins of this algorithm can be traced back to the end of the 1950s but it was not a recognised scientific field by then due to the lack of sturdy computers at that time [71]. Obviously, EAs follow Darwin's theory and mimic the natural evolution of a "population" through generations [146, 64]. A population includes a given number of individuals. Each individual is characterised by a sequence of genes. Traditionally, an individual is a candidate solution of the optimisation problem (we will see in Chapter 4 that this is not true in the case of Parisian Evolution). The population evolves toward a better adaptation to its environment (the search space) using the principles of natural selection. It is worth pointing out that the inspiration for this algorithm comes from the phenomenon called "the survival of the fittest". Thus, the individuals are evaluated: the stronger individuals are more likely to survive and transmit their genes to their offspring, whereas the weaker individuals will slake (die out). This evaluation is based on a fitness function (sometimes called a cost function). Each individual is associated with a fitness value that affects the selection mechanism to ensure the survival of the fittest individuals in a competitive environment [146]. Note that mutations of genes can occur to preserve the diversity of the population to better explore the search space. Figure 3.2 illustrates how most traditional EAs can be modelled [30]. The main steps of an EA are described in the subsections below.

There are different ways to implement an EA:

Simple genetic algorithm (non-overlapping population): Non-overlapping population is used in simple genetic algorithms. This is the "generational" implementation, which is the traditional approach, where the whole population is changed by new individuals. The evolution of the genetic algorithm relies on the genetic operators (selection, crossover, mutation). In its naive form the generational approach does not guarantee that the best individual in one generation is present in the next generation. To address this deficiency, "elitism" may be added (the best individuals of a generation are replicated in the next generation). The algorithm may quickly converge due to the operation of maintaining the best individual(s) during the evolution. To avoid premature convergence toward a local optimum, diversity mechanisms are required to cover the entire search space of a problem.

Steady-state genetic algorithm (overlapping population): There is no proper notion of generation in steady-state genetic algorithms. At each iteration of this algorithm, a portion of the population is replaced by newly generated individuals (some of the worst parents by the best offspring). The steady-state genetic algorithm follows the principle of overlapping population. The algorithm replaces a few individuals of the population on each iteration. The key again is how to select the optimal genetic operators suitable for the specific problem. For example, if the selection operator favours the best individuals, premature convergence may also occur.

3.2.1 Solution encoding and initialisation

EAs rely on randomised operators that work on a set of candidate possible solutions (individuals) in the search space [71]. These individuals correspond to a set of prospective solutions called a "population". Individuals may be encoded in various ways depending on the problem to be solved [146, 163, 17, 91]:

Binary String and Gray Encoding: Every individual is represented as sequence of $0s$ and $1s$. This is the model used in the traditional "genetic algorithm".

Value Encoding: Each individual is symbolised as a sequence of diverse values such as integer, real, char and object, which depends on the specific problem.

Real (floating-point) Encoding: Every gene of an individual has a real value, e.g. 0.1, 1.9,. . . , etc. EAs using this model are called "real-valued genetic algorithm".

Permutation or Integer Encoding: In these encoding models, each individual is symbolised as sequence of integer numbers. However, permutation encoding represents a position in a sequence. It is useful with ordering problems such as the travelling salesman problem (TSP).

Tree encoding: Every individual is a tree of objects. It is primarily used with problems that need to utilise genetic programming [66, 65, 110, 113].

Figure 3.3 shows the above five generic encoding schemes for EAs.

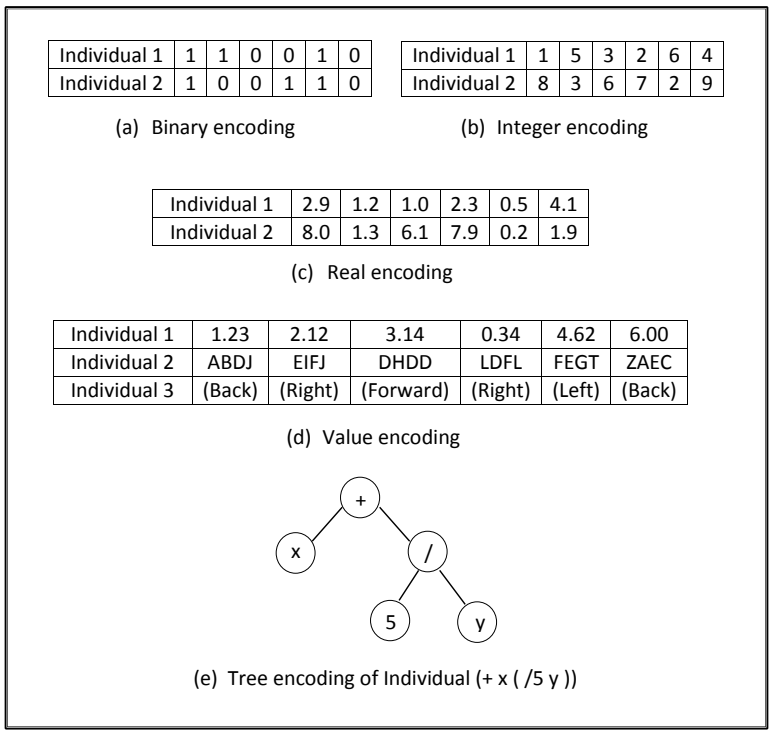


Figure 3.3: Examples of individual encoding.

3.2.2 Evaluation (Fitness function)

A fitness function is problem-dependent. The fitness function is used to evaluate the individuals. This is probably the most significant step in EAs. Fundamentally, the fitness function measures the quality of the solutions [64]. It helps to maintain the population of individuals [146, 172] as it is utilised during the selection step to determine whether an individual will more likely survive or die [92]. The fitness function does not have a closed mathematical expression and it may have many forms. For example, it may be given by the simulation of a real physical system [31]. In this case, it can be used to minimise an error between the target and the values outputted by the individuals during the generations [38, 158].

3.2.3 Selection operator

There are various selection schemes. They are used to filter the individuals (parents) in order to reproduce (breed) for selective parents and generate new offspring for the next generation [146]. The most popular selection methods are discussed below [92]. Although the selection can be performed in many different ways, it is usually guaranteed to select the best individuals, which have an elevated probability of selection. The selection process relies heavily on the use of the fitness function [71, 30]. If maximisation is considered, then individuals with the highest fitness will be more likely to be selected. If minimisation is considered, then the ones with the lowest fitness will be more likely to be selected.

Proportional selection or roulette wheel selection is a way to randomly select any individuals. It is the simplest and most common technique. The motivation of this approach is that individuals with the highest fitness values have the greatest chance to survive and be selected as parents [113]. The probability of selection of individual i is:

$$P(i) = \frac{fitness(i)^{\alpha}}{\sum_{k=1}^{n} fitness(k)^{\alpha}} \tag{3.1}$$

with n the population size (i.e. the number of individuals in the population), with α a control parameter. The algorithm below shows the main principles of this selection:

1. Calculate the fitness value of all the individuals of the population and store them in vector **A** of n elements.

2. Compute a new vector **B** of n elements, where each element is the accumulative summation of fitness values of symmetric elements and all previous ones in vector **A**.

3. Generate a random number x from the given population interval (the range of values in **B**).

4. Select an individual i when x is in the range of B_i, where $i = 1, \ldots, n$.

Rank-based selection: This technique sorts the individuals in the population into a list which displays the individuals from the best to worst depending on their fitness values. Each individual is associated with a certain rank. Therefore, the selection mechanism is based on the rank of individuals rather than alteration in fitness values. The rank of the best individuals have a higher probability to be selected than those of the worse individuals.

Tournament selection: This method relies on holding a competition of randomly chosen individuals. The number of individuals involved in each selection iteration is called the "tournament size". The best individual (in term of fitness) among them will be the selected individual.

Elitist strategy: This is an addition to other EA selection methods. This heuristic method forces EAs to carry forward a fixed number of the best individuals of the current population into the next generation [33].

3.2.4 Crossover operator

This is an important operator in EAs. Its probability to occur (P_c) usually ranges between 0.6 to 1.0 [17]. Pairs of individuals are randomly selected to produce two new individuals. The two parents swap some of their genes depending on crossover points. To determined these points, there are several types of crossover mechanism [5]. Figure 3.4 presents different types of crossover operator.

Single-point crossover (1x): In this operator, one position between the pairs of individuals is randomly chosen. Then, all genes after that position are exchanged between the individuals. The selected position can be among any gene position from the first through the last gene. This operator is very fast, but it has the problem of decreasing diversity especially when some individuals are similar in the population.

Two-points crossover (2x): In this operator, two crossover sites are chosen randomly. After that, the sequence of genes between them are swapped.

Uniform crossover (Ux): This is different from above crossover operators, since the Ux does not use crossover sites, but randomly shuffles the genes of the parents, by the probability (P_c) of exchanging genes, in order to create two children [111].

Tree-encoding crossover: This is similar to a single point crossover in terms of selecting a random crossover point somewhere in the tree of the parents. On the other hand, this type of crossover exchanges the part of tree below the crossover point between the parents to produce new children [113].

3.2.5 Mutation operator

This is used to randomly change one or more gene values on an individual. The number of alternative genes depends on the mutation rate (P_m), which is ordinarily between 0.001 to 0.05 [17]. Basically, mutation influences the whole population due to boosting the diversity within it. This is done by generating new individuals through the

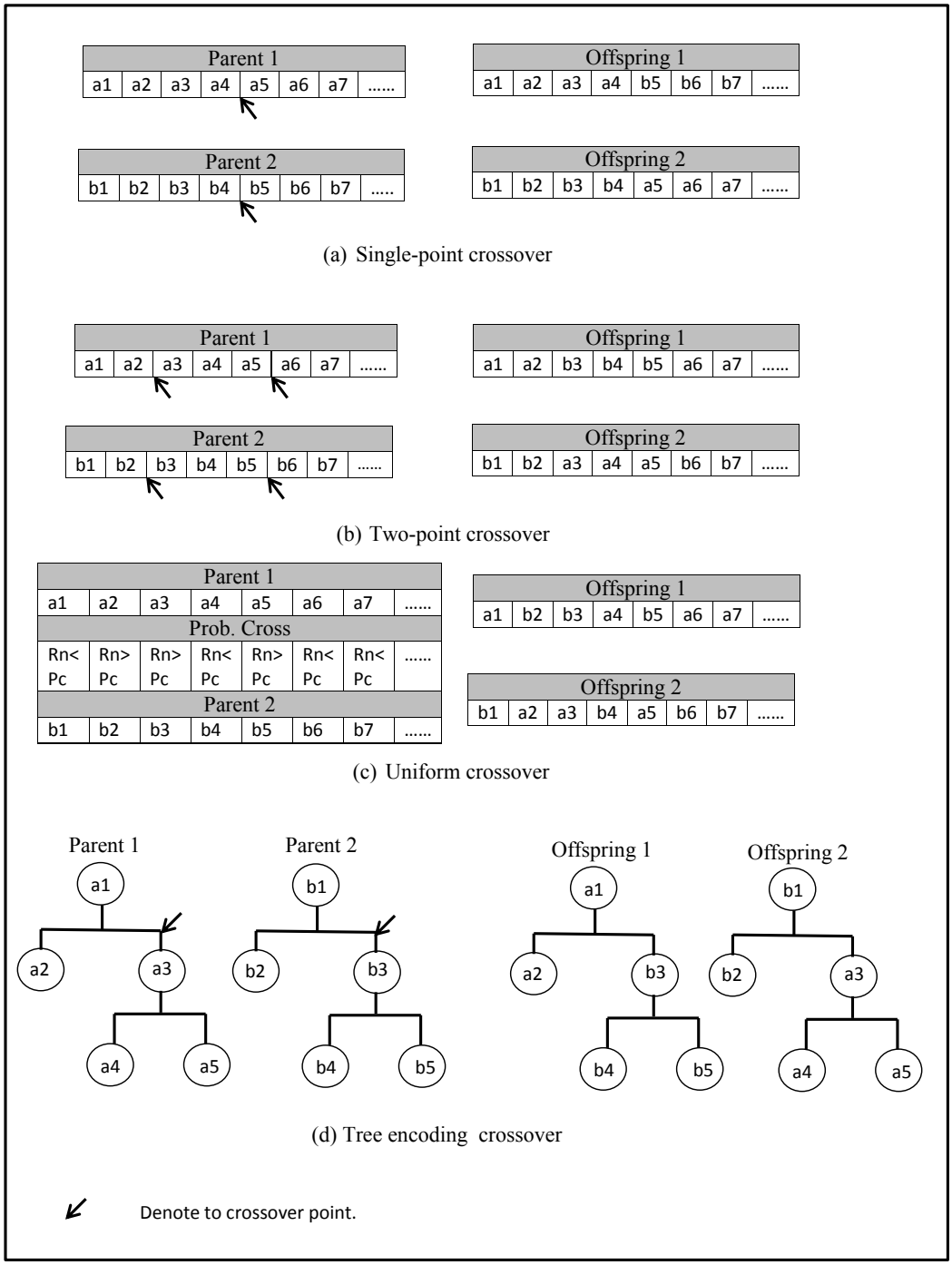


Figure 3.4: Different techniques of the crossover operator.

generations. Therefore, it prevents stagnation in the convergence of the optimisation technique. There are different types of mutation operator (see Figure 3.5) [5]:

Flip Bit (Single-point) mutation: A single gene value in the individual at a selected mutation point is changed into other value in the range of this gene. When the gene has a binary encoding, the value is flipped from "1" to "0" and *vice versa*.

Uniform mutation: This is a classic technique in which each gene of an individual has an equal likelihood to be mutated by any value in the solution space [44].

Boundary mutation: This is a special case of uniform mutation. It is used only with floating-point encoding. The newly generated allele z_i is either the upper bound (UB) or the lower bound (LB) of the domain, with equal probability [99]. i.e.,

$$z_i = \begin{cases} \text{UB} & if\, x > 0.5. \\ \text{LB} & if\, x \leq 0.5. \end{cases} \tag{3.2}$$

Where x is a random number between 0 and 1.

Tree encoding mutation: This mutates selected nodes of the tree, with a new sub tree, to create new offspring [113].

3.2.6 Termination

EAs repeatedly execute the operators presented above on each iteration (generation) to produce a new generation. The iterative nature of this algorithm helps to increase the fitness of the best individuals in each generation towards the global optimum [17]. The termination of EAs relies on different possible stopping conditions:

- Determining a fixed number of generations.
- Extracting the problem solution: the individual with the best fitness value (or proximity to certain conditions, e.g. a threshold, which serves minimum criteria).

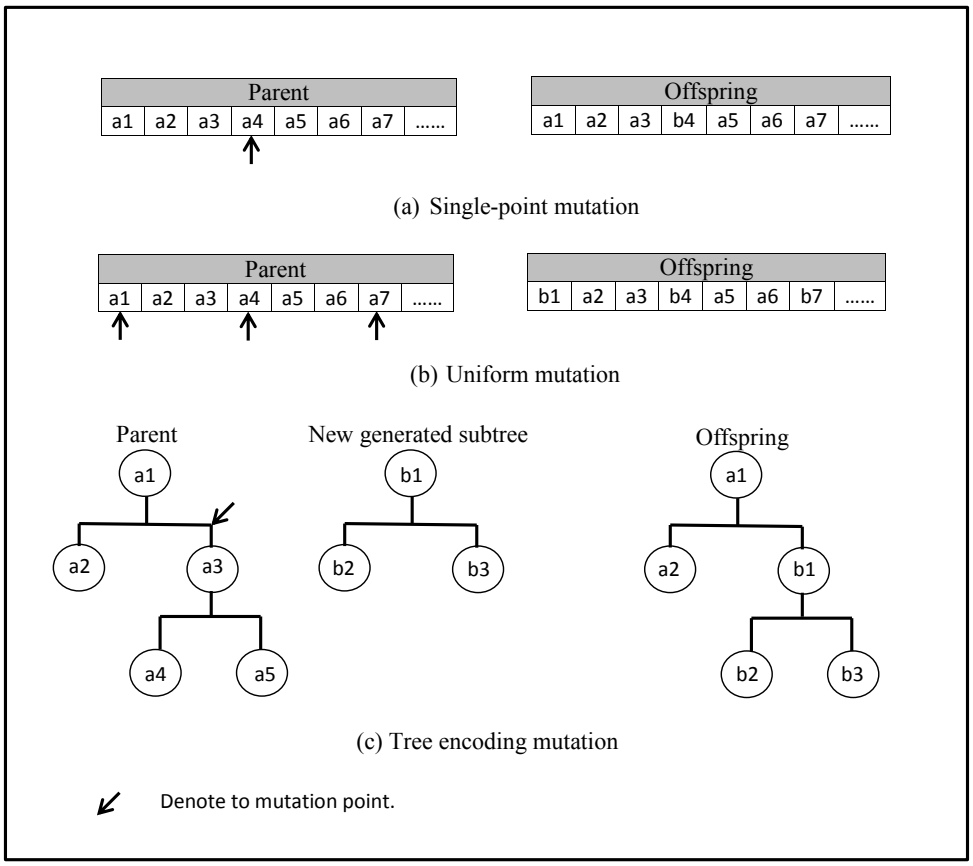


Figure 3.5: Different techniques of the mutation operator.

- "Allocated budget (computation time/money)" that should not be exceeded.

- Manual monitoring or recording the level of similarity in the population of individuals. i.e., a vast number of individuals have congruous values at most positions [146, 92].

3.3 Evolutionary algorithms and their applications to image processing

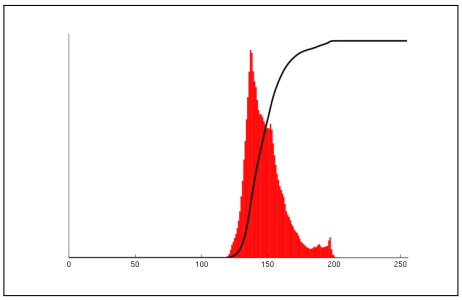
This section reviews the application of EAs in image processing. The section is divided into three parts. The first subsection is about image enhancement, which acts as a pre-processing of image data before any application of image processing using EAs. The second subsection discusses different approaches to image segmentation using AE. Subsection 3.3.3 presents a method used in 3-D imaging: stereo vision and 3-D tomographic reconstruction in nuclear medicine.

3.3.1 Image enhancement

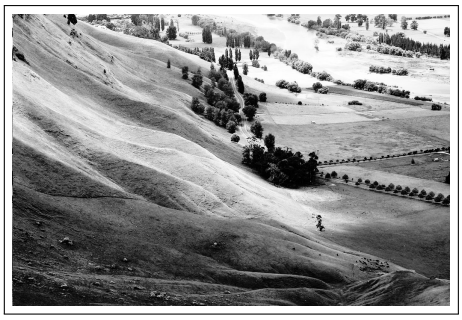
Image enhancement is the process of improving image quality with compatible features [112]. It enhances images to provide a better understanding and visual perception [130] , reveal the implicit data on the images and change image attributes to make them more appropriate for other automated image processing applications [77, 142]. Image enhancement is strongly required due to the limitations of the hardware used to captured the images, dynamic light conditions and changing captured environments. It is used in different fields such as computer vision, remote sensing, robot navigation, medical image analysis and satellite image analysis.

Histogram equalisation (HE) is one of the most simple, effective and popular approaches to enhancing image quality [67, 73, 142, 3]. The objective behind this method is to display a uniformly distributed histogram as a result of computing the cumulative intensity function of the input image. It is used to increase the global contrast of numerous

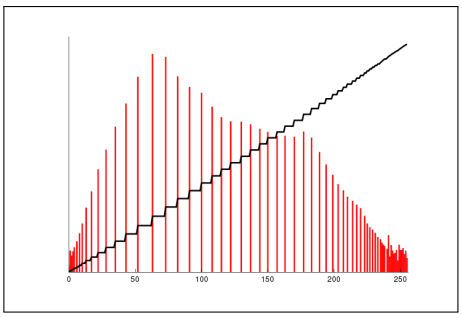
(a) An unequalized image



(b) Corresponding histogram (red) and cumulative histogram (black)



(c) The same image after histogram equalization



(d) Corresponding histogram (red) and cumulative histogram (black)

Figure 3.6: Enhancement image process using Histogram Equalisation (HE).

images. This method tries to distribute intensity image values almost equally across the range of an image [35]. Figure 3.6 shows improving an image's contrast. Many HE methods have been proposed, such as Global Histogram Equalisation (GHE) [88], Local Histogram Equalisation (LHE) [73], Adaptive Histogram Equalisation (AHE) [149], Minimum Mean Brightness Error Bi- Histogram Equalisation [35] and multi-peak Generalised Histogram Equalisation (multi-peak GHE) [36](see [142, 73] for other versions of HE).

Although there are many HE approaches, to date no particular method suitable for all problems has been found. It takes a long time on a complex search space [115, 114], since most classical approaches depend on statistical techniques [78, 20]. For this reason, EAs have been receiving great interest to tackle this as an optimisation problem in order to achieve natural contrast images with lower processing time [125], unlike the traditional enhancement methods in which the user spends a lot of time in the supervision of each image [114]. Below we briefly introduce the diverse types of image enhancement approaches based on EAs.

Munteanu and his colleagues [114] integrate a local enhanced method with a genetic algorithm (GA). They use AHE as an example of a local enhancement method where each pixel is assigned a value according to a histogram equalisation transform performed in the $n \times n$ neighbourhood of that pixel. They use the GA to tune the parameters that control the contrast of the images, relying on the unique fitness function of their GA. This function exploits the idea that a good contrast image has a high number of "edgels", which means pixels centralised on a edge. Therefore, the Sobel edge detector is boosted in their fitness function. A new mutation (Principal Components Analysis mutation (PCA-mutation)) is adopted. Principal Components Analysis (PCA) is used to highlight what components are important and which ones are negligible. The mutation is favouring the homogeneity of the components to ensure the diversity in the genetics of the population.

In a similar contribution to Munteanu, Hashemi and his colleagues [73] present a contrast enhancement method based on a GA. They also use a Sobel edge detector in their fitness function. Low dynamic-range grey and colour images are used in their experiments. The structure of each individual is a vector of random integer numbers (between 0 and 2^{n-1}) sorted in ascending order. The length n of the vector depends on the number of grey levels in the input image. This structure is utilised for remapping the grey level values in the input image for the sake of boosting the convergence speed in order to get to the optimum solution. Roulette wheel selection, two point crossover, and single point mutation are used.

Naoum and Al-sabbah [115] apply steady-state genetic algorithm (SSGA) to enhance colour images. The rely on the HSI colour model. Tournament and Roulette Wheel selection, two point crossover, and one point mutation are used with SSGA. The fitness function combines Mean absolute error (MAE) and Peak signal-to-noise ratio (PSNR). The setting of (P_c) and (P_m) are tuned automatically through the evolution. The final testing phase makes use of both Dice Similarity Coefficient (DSC) and partial Hausdorff distance for evaluating the performance of their enhancement algorithm.

Since now, hybridisation among various optimisation methods have been widely applied in visual image enhancement field. Hoseini and Shayesteh [78] propose a hybrid method working in parallel to achieve high disparity and naturality of gray scale images.

Figure 3.7: Results of different enhancement techniques: (a) original, (b) Hoseini et al. method, (c) histogram equalisation, (d) linear contrast stretching, (e) fuzzy method (Image from [78], courtesy of Pourya Hoseini).

These methods are Ant Colony Optimisation (ACO), GA, and SA. The ACO is used to modelling a transfer function which generally relies on global intensity transformation. Then, GA is utilised to tune the ACO parameters. For this purpose, their paper depends on a fitness function that includes three components, which are the global standard deviation of intensities, the global entropy of gray level, and Sobel edge detection. They use SA to increase convergence time of ACO. Their results are compared with other popular image enhancement (IE) methods such as linear contrast stretching, HE, and fuzzy logic. As a result, the proposed technique shows improvement according to a human observer. Figure 3.7 shows the efficiency of the proposed method that applies on images with lower resolution.

Enhancing a retrieved watermark image, which is embedded within an original image, based on GA was introduced by Shih *et al.* [141]. To solve the problem, they start with random initial population of individuals. Each individual consists of 64 genes of binary number. The individuals are evaluated with fitness function f_i that computes the summation of the absolute difference between the embedded, $Watermark_i^{\alpha}$, and extracted, $Watermark_i^{\beta}$, watermarks. For reproduction individuals, The authors apply both one-point and two-points crossover and a mutation operator.

3.3.2 Image segmentation

Image segmentation (discrimination) is the process of subdividing an image into multiple, non-overlapping regions or objects that are considered homogeneous portions, having attributes of either a similar texture or colour. The task of image segmentation

is still one of the most difficult ("ill-defined") and unsolved problems in the image processing field [67, 139, 19]. Image segmentation is an essential task in many imagery applications, including object detection, feature extraction, object recognition and object classification [176]. So far, many segmentation methods have been proposed but none has satisfactory partition results: There is no general and unique method found across the different types of digital image processing techniques. Generally, each segmentation method is based on a set of control parameters, which are difficult, if not impossible, to tune in some imagery applications even if these parameters are fit for another particular application with a specific type of images [174, 176]. Therefore, EAs can be used to optimise the relevant parameters in certain existing segmentation algorithms [20] in both grey levels and multi-component images, which was verified practically by Chabrier *et al.* [32]. In addition, the interaction of EAs toward the application of image segmentation can be categorised into two main groups:

1. Hybrid (adaptive) image segmentation applications where the genetic algorithm aims to tune the parameters of segmentation methods to improve the final results of such a segmentation method [32], and

2. GA-based segmentation where AE plays a major role in labeling growing image's regions by using the principle behind pixel-level segmentation [55].

In both groups, the image is segmented according to various criteria (e.g., grey level, colour, or texture on an image) [132].

Ramos and Muge [132] present an adaptive segmentation method that combines k-means (an unsupervised clustering approach) with a GA. Their motivation is to detect the optimal colour cluster regions in a complex colour feature space image, which has different types of textures, such as ornamental stones, human skin marks and colour maps, by minimising the internal feature distinction among the same colour clusters. However, maximising the feature variances between various colour clusters was taken into consideration.

In another example where GAs are significantly used, Bhanu *et al.* [19] apply an adaptive technique using a GA to improve the performance of partition criteria in outdoor colour imagery. They employ an algorithm to adjust 14 control parameters of the Phoenix

segmentation procedure. This is done to be valid for many images with different characteristics as well as various scenes of CV applications, such as instance image classification and image recognition. Hence, the authors design the hybrid segmentation method to be a part of the image recognition stages.

Another fundamental work in which GA is used in a region growing segmentation approach is by Feitosa and colleagues [57]. They employ a supervised segmentation method. This method attempts to adjust the segmentation parameters according to comparing the similarity degree of the segmentation regions' results with a template of the image segmentation objects supplied by the user. In particular, they try to minimise a fitness function that scales the weighted heterogeneity of merging reference objects with the segmentation results through the large intersection space.

Zanaty and Ghiduk [172] proposed an innovative segmentation technique to solve the same problem (region growing segmentation technique). Their method combines a classical GA with a seed region growing method to overcome the over-segmentation problem. Figure 3.8 shows the accuracy of the segmentation results when their algorithm is applied to magnetic resonance image (MRI) images. It is worth pointing out that the contributions of this article are:

1. It presents a new method that specifically works on 3-D medical images, MRI datasets with weak boundaries.

2. Their algorithm starts with a random population distributed all over the image.

3. The structure of a chromosome contains three parts: control genes, grey-level genes, and position genes.

4. Chromosomes are represented in binary forms of 0s and 1s.

5. It hands out a new objective function and individuals' representation that are aimed to improve the image segmentation.

6. It evaluates and compares the final results of this work with the fuzzy C-means clustering (FCM) method.

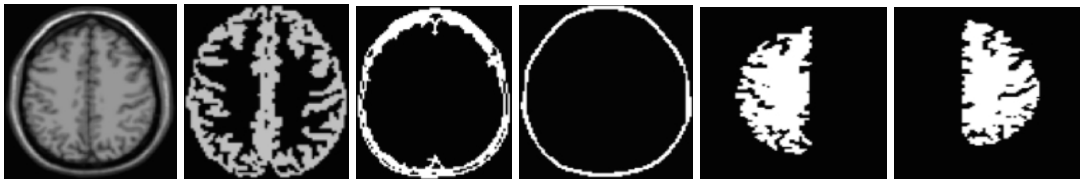


Figure 3.8: Results of genetic algorithms and the region growing method for segmentation MRI (Image from [172], courtesy of Zanaty).

As a result, the proposed technique provides results better than the FCM method.

Another fundamental approach of using GA for 3-D image segmentation was aforementioned by Zanaty *et al.*; Pignalberi and his colleagues [129] concentrated on range images (RIs). Such images are coloured according to the distance from the sensor that scans the image. In fact, each pixel in a RI indicates the value of the distance from the sensor to the foreground object point. They attempt to optimise a collection of parameters that are often utilised with a RI segmentation algorithm in order to distinguish the outside surfaces of 3-D objects.

In an implementation of pixel-level unsupervised segmentation, Peng *et al.* [127] propose a novel hierarchical distributed genetic algorithm, which produces a parallel image segmentation without a prior hypothesis of references of segmentation images excluding the number of partition districts. It is important to note that the researchers work on gray-scale images for segmentation purposes.

As a part of the unsupervised paradigm and far away from any manual segmentation intervention, Gong and Yang [65] applied an unsupervised segmentation system that is more complex for colour images. They use a hybrid framework that integrates a neighbourhood-based segmentation method and GA. In fact, they used the final algorithm in a two-pass process. In the first pass, GA is used to minimise a fitness function, which is produced using the Markov Random Fields (MRFs) method. The aim of this stage is to find the histogram of the original image. The advantage of the histogram is to find the location of peaks, which is necessary to detect the cluster information. Therefore, GA comes to optimise the results of labelling the pixels in which the closest cluster belongs. This is done by using the Euclidean distance as a fitness function of GA. Second, GA was utilised to optimise the final segmentation. It is important to note, for this implementation, that the researchers use a quad-trees structure to represent the individuals of the GA population.

Zingaretti *et al.* [176] propose a method similar to Gong and Yong's idea of defining a two-pass GA. However, they generally rely on a histogram-based thresholding algorithm (instead of the neighbourhood-based segmentation method) after the system has divided the input images into two "chromatic and achromatic" regions and processed each one separately.

Cheng and Gong [37] propose a more robust adaptive algorithm that combines three approaches: the FCM method, ACO and GA, which are strong procedures in the image segmentation and combinatorial optimisation field. In the first part of this hybrid method, the FCM aims to find the shortest path from various image patterns to a cluster centre in order to cut off the images. After that, ACO and GA work in parallel to improve the "efficiency of clustering", accelerate the convergence of clustering, and escape from entrapment in local minima. As a result, this method provides a high accuracy level on image segmentation, especially on detecting "fuzzy and exiguous edges".

For more reviews on unsupervised image segmentation methods, Zhang *et al.* [174] present a useful survey paper that demonstrates and discusses an analytical point of view on various numbers of the popular image segmentation mechanisms.

3.3.3 Image reconstruction

Image reconstruction is widely used in medicine for 3-D tomography reconstruction (in CT, CBCT, SPECT, PET, MRI, and 3D-ultrasound (US)) [52, 155, 156] (see Section 4.3.2), computer vision (stereo vision and 3-D reconstruction from multiple images) [70], robotics (simultaneous localisation and mapping (SLAM)) [103], etc. Image reconstruction is an inverse problem: an unknown image has to be produced form a set of known projections. The nature of the unknown image and of the projections is problem-dependent (see Figures 5.4 and 5.5 for examples of projections and reconstruction in tomography). Image reconstruction is often ill-posed in real applications: a solution does not necessarily exist (e.g. in case of excessive noise), and the solution may not be unique. This problem can be solved as an optimisation problem, and in such cases, evolutionary algorithms have been shown to be generally efficient [161].

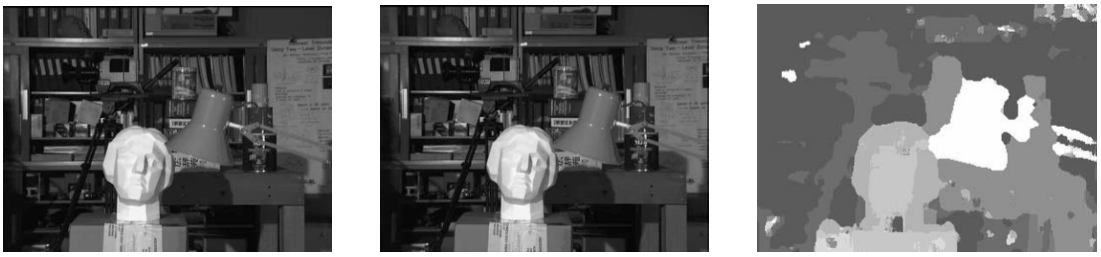
In this section, we will consider stereo vision systems based on a classical EA. In the next chapter, we will discuss stereo vision systems based on the Fly algorithm and a more recent application (3-D PET reconstruction in nuclear medicine).

In stereo vision systems, the main task corresponds to producing a 3-D model of a scene given pairs of 2-D images: right and left images (see Figure 3.9a and 3.9b). It is a difficult and important problem in CV. The stereo vision problem has become an active subject in different image application areas, such as object recognition, inspection, manipulation, stereo sequence coding, intermediate view generation, robotics, virtual reality, motion estimation, and entertainment [144, 71, 168]. In fact, such systems aim to mimic the human visual system (HVS), which is automatically capable of finding out the depth information from a 3-D constructional information. This information is received by a pair of eyes, which distinguishes little variations in the image of each eye due to their distinct positions (see Figure 3.10). A stereo vision system accomplishes such simulation of HVS by using two or more cameras and a computer. The main objective of this system is to reconstruct the depth of the objects and define a smooth disparity map of the 2-D world in order to determine corresponding points among images using the stereo matching technique. In particular, this method aims to address the explicit inequalities among the images to all common points. It relies on the principle that the corresponding points in the different images are projected on the same point in the 3-D scene. The disparity map, (see Figure 3.9c), is formed as a result of two or more images [66, 71, 84, 70], which have slight differences aside from the many common features, by using either of the following two techniques:

1. Using two or more digitally calibrated cameras, which are separated by a known aligned distance between each other.

2. Using only one camera to capture different images from several positions [128, 168, 43].

Figure 3.10 shows the epipolar geometry of a stereo vision system or binocular stereo method. In [71], more detail about the epipolar geometry of the stereo vision system is provided.



(a) Right image. **(b)** Left image. **(c)** Disparity map.

Figure 3.9: . Stereo pairs (right and left images) from the Multiview Image Database, University of Tsukuba and the disparity map of them (images from [60], courtesy of Andrea Fusiello).

L is the distance between the two cameras on the baseline; x is the depth, which is calculated using equation 3.3, f is the focal length of the camera; and $d_l + d_r$ is the disparity map of the corresponding points.

$$x = \frac{f(L + d_l + d_r)}{d_l + d_r} = \frac{fL}{d_l + d_r} + f \tag{3.3}$$

Up to now, many stereo matching techniques have been introduced. Generally, the present approaches belong to two main classes: either feature-based methods or area-based methods. The former rely on revealing features in the first image and then attempting to identify and match these features in the second image. Such methods depend on calculating features, such as zero-crossing points, edges, corners and line segments. This method estimates the depth information accurately, but it needs a complex interpolation process to get a complete disparity map. Area or intensity-based approaches work on computing a disparity map for each pixel in the image. The core of this algorithm is that match neighbouring pixel value within a window, which is a block of $n \times n$ pixels, between images. The intensity-based methods evaluate the disparity of the pixel according to local neighbouring information. Therefore, this method is based on two facts. First, it is based on the disparity of the pixel influence by the disparity of its neighbouring pixels. Thus, the window's size has a big role in finding the corresponding points. The optimal window size should be estimated in order to avert disparity variation with a large window and to include as much intensity variation as possible for reliable matching with a small enough window size. Second, the intensity-based methods need additional techniques, which are used to compute local information and are categorised as follows: cross-correlation, least-squares region, etc. [71, 43, 135, 66].

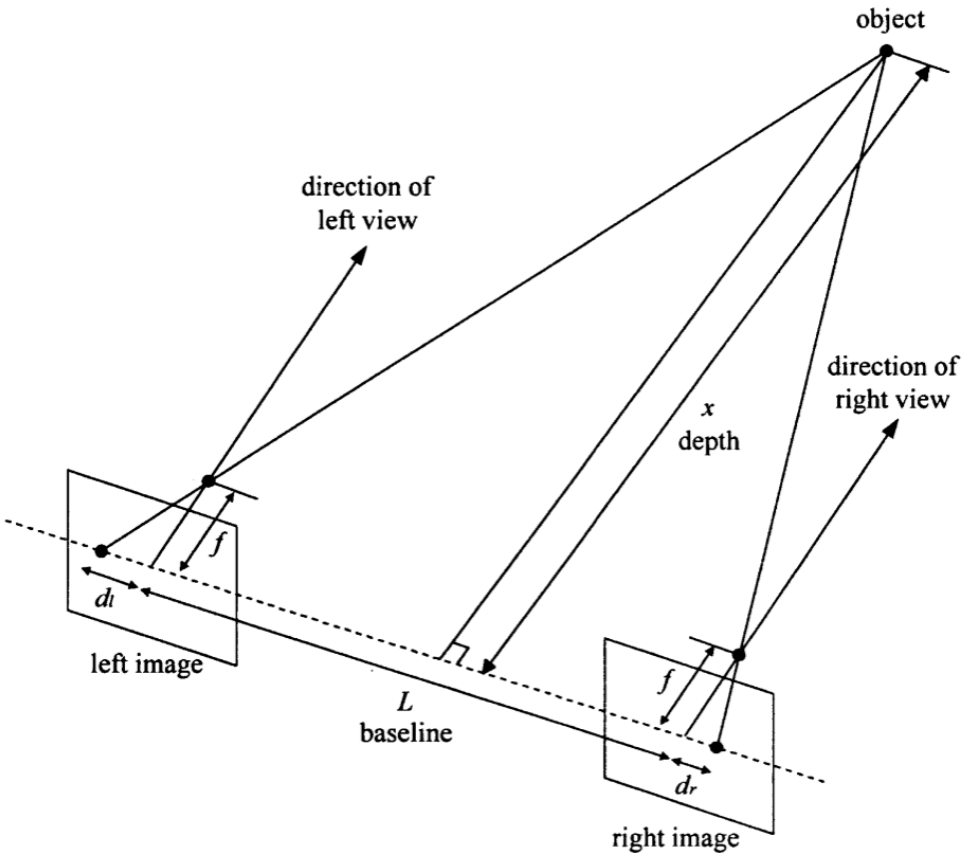


Figure 3.10: Epipolar geometry of a binocular stereo system (image from [70], courtesy of Hafezi).

Most of the existing stereo matching methods aim to match the points that have the same neighbourhood attributes in both images [46]. Estimating the disparity map of each image pixel leads to an expensive run time in such methods. GAs appear to be competitive candidates to tackle this problem; a large amount of data (pixels in the images) to find a global minimum disparity map at each pixel position. Thus, the stereo matching problem can be formalised as an optimisation problem [168, 102, 66, 137], and GAs can be applied to solve it in the context of a stereo vision system. As reported by Saito and Mori [135], each pixel has many candidate values of disparity based on various window sizes. They rely on the sum of squared differences (SSD) for a specific window size to calculate the disparity map between a pair of images. Then, the GA is utilised to evaluate each block, determining the optimal disparity map in each pixel. The fitness function is designed to find the possible smallest error between the images. It relies on the similarities between corresponding points and the continuity of the disparity map. This presumption may not encompass the noisy images or the part of the view that has plain colour [66]. In the same way as Saito and Mori, Dai and two other researchers apply adaptive GA to stereo matching on different databases, which

are represented by Tsukuba stereo images [43]. In contrast, Han *et al.* [71] depend on a fixed window size to calculate the stereo correspondence problem. Before that, the modified nonlinear Laplace (MNL) filter is applied to extract the region parts from the reference images. This filter is greatly affected by the generation of the disparity map. Next, GA is used with unique representation of individuals that use a 2-D block structure instead of a one-dimensional (1-D) to find out the best disparity map between the images. Using the same main ideas as Han's paper, Hafezi *et al.* [70] propose a new stereo correspondence system structure with a fixed-size window. This structure uses GA with respect to implementing SSD as a fitness function. However, it relies on rank transform to produce the initial disparity map for each single pixel. Another study by Gong and Yang [66] uses GA to solve the problem of stereo matching, inspiring their idea of using MRFs to shape the fitness function. Moreover, the researchers depend on a cooperative genetic algorithm, which takes into consideration not only the pixels on an image but also the neighbours. Therefore, they adapt the genetic algorithm to include the cooperative principle. This is done by using a quad-tree structure, which ensures the inclusion of the pixels and their neighbours for each individual of the initial population. For this special representation of individuals, they depend on a different crossover operator, which is called graft crossover, as well as three methods of a mutation operator, which are named splitting, merging and alteration. In 2009, Zhang *et al*. [175] combine pyramid division, GA, and propagation stratagem to generate a final disparity map in a robust stereo vision matching algorithm. Other studies have considered different techniques of capturing the stereo images by using one camera and a laser beam instead of using two cameras to achieve the stereo correspondence problem, such as in [46, 168].

3.4 Challenges with evolutionary algorithms

As a matter of fact, finding an optimal solution is difficult in image processing applications due to several factors. Such problems are often ill-posed inverse problems. They can be affected, for example, by poor spatial resolution, changing environmental conditions (e.g., lighting), and image format (i.e., images of the same captured view will look different because they may be taken using different cameras, various angles,

distances and sensor sensitivities). This chapter has highlighted various research papers in evolutionary computing and its applications. Focus has been given to imaging applications. We saw that artificial evolution can be used as a black-box optimisation tool, however, problem-dependent implementations provide better results [125]. Thus, it is difficult to develop a universal and robust method that adapts to all the problems in imaging.

EAs have a significant impact on solving complicated problems as optimisation problems due to their unsupervised nature. EAs can be used when little is known about the problem to optimise [157]. It is usually not a good strategy to consider EA as a “blind" method because one may lose the chance to adapt to the problem. It is indeed possible to adapt the evolutionary mechanisms to the specificities of the problem [157] or to rely on domain-specific knowledge [166]. It helps to improve the efficiency of the algorithm and reduce its computation time. Complicated problems will help to compare customised EAs to “pure” classical and black-box optimisation methods to assess the advantages provided by the “intelligence” set in the genetic operators [157].

Although EA is a search method that easily adapts to complex search problems because of the parallel exploration of the search space, EAs are faced with different challenges in image processing applications. These challenges are [124]:

1. Many researchers consider EA to be a slow black-box optimisation method, which does not use relevant domain knowledge. However, it has been demonstrated many times that this judgement is ill-conceived. EA can be implemented more cleverly to include domain knowledge, and it can be extremely competitive when solving hard problems.

2. EAs may require a lot of computer resources to run simulations each time a fitness function is calculated. The final result may depend on the parameters of the algorithm, such as the number of individuals, probability of the genetic operators and the type of selection. However, to reduce the pressure on computer’s resources, the user must observe and decide how changing these parameters will affect the final results of the problem. Note that varying population sizes and self-adaptations can be used to tackle this issue [155].

3. Another fault of EA is that an optimal solution is not guaranteed in a finite time. It completely falls into the category called "Generate-and-Test". This affects the user's decision of what the best and worst case scenarios are through the constant optimisation run times. A possible solution is to boost computer processing power as well as parallelise the implementation of the EA. Note that this is the case of most optimisation methods anyway.

4. Occasionally, due to the random nature of the method, to find a solution through evolution, the method may hardly converge toward an optimal solution (e.g., if the parameters of the genetic operators are not well set).

5. EAs fall in the local optimal point; they are less likely to stop at the local optima than classical deterministic optimisation methods, which may not have the ability to reach to the global optimal point [157]. This is because they rely on the crossover operator, which decreases the genetic diversity of the population. The solution is more likely to use the mutation operator to increase the diversity of the individuals. Other diversity mechanisms (such as "new blood" or "immigratiom") can also be introduced. The difficulty is to choose or design all these genetic operators that lead to a global optima is a minimum of time.

Chapter 4

Background: Parisian Evolution

In this chapter we present an overview of a special type of CCEAs based on the Parisian approach: the Fly algorithm. The main differences between CCEA and classical EA are explained. We present various applications of the Fly algorithm. We explicit the novelty of the proposed in this thesis with respect to the previous implementations of the Fly algorithm.

4.1 Cooperative co-evolution algorithms

A CCEA is a specific form of EA. Over the last few decades, CCEAs have been designed to emulate a natural population that establishes a solution by a collection of individuals working together. For example, CCEAs can be used to simplify a complex search space by splitting it into a number of smaller sub-spaces [162]. Co-evolutionary techniques provide an effective manner to deal with large, complex problems using the "divide-and-conquer" strategy. This technique can be implemented to find a global solution by gathering the partial solutions of individuals on behalf of the best individual, i.e. each and every individual is treated as a part of the problem solution [136, 126, 123, 29]. Based on this concept CCEAs are formulated from EAs in terms of using the rules of artificial Darwinism, i.e. classical evolution-style genetic operators, such as selection, crossover and mutation. However, the former type of algorithm adopts this cooperative principle by building a fitness function that considers each individual and its neighbourhood [26, 167]. Instead of using one fitness function to assess an individual, as in the traditional EA algorithm, two fitness functions can be used in CCEA:

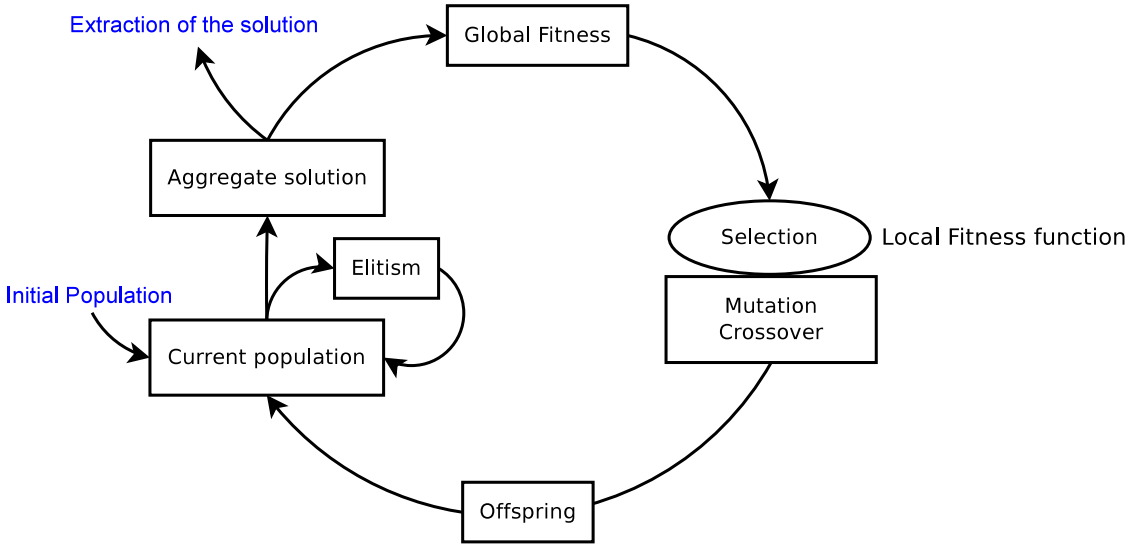


Figure 4.1: Principles of the Parisian approach (including the Fly algorithm).

- local fitness, which evaluates an individual from the population as a part of the solution, and

- global fitness, which assesses the whole group of individuals obtain the final joint solution of the problem (see Figure 4.1) [136, 156, 123, 30].

In [29], each individual is evaluated based on its collaborators. One of the main objectives behind a CCEA algorithms is to reduce the computational pressure of the genetic search process by applying parallel EAs to find an optimal solution [136, 167].

4.2 Parisian approach

An example of a CCEA is the Parisian EA. The solution of this algorithm is embedded within a set of individuals instead of one, as in classical AE. Individuals assemble together to build a significant solution to the problem, driving the whole population into attractive areas of the search space, where an ideal solution is likely to be found.

This approach has the benefit of co-evolution approaches: a population works together, no matter whether the individuals are identical to or distinct from each other. The Parisian scheme tries to maintain the diversity of the population, which makes such an algorithm slightly more complex than the traditional evolutionary algorithm. However,

the intrinsic parallelism of the Parisian approach reduces the computational waste occurring in classical EA by keeping the whole population as the optimisation problem solution rather than just the best individual [26, 103].

The main challenge is to design a fitness function that keeps a set of individuals working together to raise up the solution of an optimisation problem to the global solution. Thus, the Parisian approach contains the features of traditional components of EA, but it has the following unique features.

- Two fitness functions: "global" fitness, which is computed on the whole population, and "local" fitness, which is computed to assess the contributions of each individual to the global solution. Global fitness may be the sum (or a complex combination) of local fitnesses. The local fitness function is used during the selection process. The global fitness function is the function that the algorithm will maximise (or minimise). However, it may not be explicitly used during the evolution phase but it can be used as a stopping criteria when stagnation is detected.

- The use of a diversity mechanism, such as a sharing scheme, to prevent the solution from falling in a specific localised area and to ensure that it has distinct individuals.

The Parisian approach has been successfully used in different applications, such as stereovision [27, 26], photogrammerty [49] and medical imaging [159, 160, 8]. Most of the applications mentioned previously rely on a Parisian algorithm called the Fly algorithm [102].

4.3 Overview of the Fly algorithm and its applications

The Fly algorithm was initially developed as a fast EA in stereovision for robotic applications, such as obstacle detection [105] and SLAM [103]. The Fly algorithm heavily relies on the Parisian approach. One of its main characteristic is that individuals correspond to extremely simple primitives: flies. Each fly is a 3-D point. Each fly is projected onto one or several image planes depending on the optimisation problem

considered. Removing an old fly and adding a new one has to be a fast process to allow it use in real-time applications.

In its original implementation, the fly is projected onto the image planes that correspond to a pair of stereo images (see Section 4.3.1 for further details). The position, orientation, and lens of the cameras are known. The pixel value corresponding to a projected fly is extracted from the pair of stereo images. For robustness, pixel neighbourhoods are used rather than single pixels. The fitness function is proportional to the differences among the pixel values between the neighbourhoods of the projected point in both images. A fly is on the surface of an obstacle, e.g. a wall, if the corresponding points on both images have little disparity between the neighbourhoods. The flies evolve using the typical steps of EAs. As a consequence, the flies will eventually gather on the surfaces of the obstacles. Detecting objects is important in SLAM as it allows autonomous robots to avoid collisions.

The first version of the algorithm is generational. In the first generation, the flies are randomly generated, forming the population of parents. Then, at the next sequential generation, the parents reproduce according to the genetic operators (selection, mutation, new blood, etc.) to generate a whole population of new individuals: the offspring. Note that crossover is not commonly used in the Fly algorithm as two good flies on different objects would lead to a bad one in between. During the selection process, the flies are evaluated based on their local fitness function(s). In turn, the offspring become parents, and the cycle continues.

Algorithm 1 shows the classical generational Fly algorithm [136] when the local fitness function needs to be maximised. The Fly algorithm can be implemented following the steady-state EA paradigm. Our implementation follows this principle. In each iteration of this algorithm, a set of individuals is replaced by newly generated individuals, i.e. a few of the worst parents are replaced by a few of the best offspring.

Algorithm 2 shows the steady-state Fly Algorithm [136]. We have used the steady-state Fly algorithm instead of the generational Fly algorithm because the former algorithm works well with complex search space, and we do not need to wait long to find the best offspring from mutated parents. The offspring are added directly to the previous

Algorithm 1 Overview of a possible implementation of a generational Fly algorithm

```
repeat
  for i = 0 to pop do
    Select genetic operator
    if Genetic operator is new blood then
      Add a randomly generated fly to population of offspring
    else
      if Genetic operator is mutation then
        Select randomly two flies f1 and f2 from the population of parents
        Compute local fitness for f1 and f2
        if fitness(f1) < fitness(f2) then
          add mutation(f2) to population of offspring
        else
          add mutation(f1) to population of offspring
        end if
      end if
    end if
    Offspring become parents
  end for
  Compute Global fitness for all population
until algorithm is convergence
Where:
pop: population size (numbers of flies).
f1 : fly f1.
f2 : fly f2.
fitness(f1): local fitness of fly f1.
fitness(f2): local fitness of fly f2.
mutation(f1): new fly created by mutation of fly f1.
mutation(f2): new fly created by mutation of fly f2.
```

population to start next the iteration. However, in the generational Fly algorithm, we need to wait to mutate all parents to get a new population of offspring which will be passed to start a new iteration. Hence, the generational Fly algorithm is computationally expensive.

Following its success in robotics, as shown in [27], the Fly algorithm has been adapted for ET in nuclear medicine, in SPECT [28], and in PET [155] and it has been used in the Lamp problem [150]. The next subsections show some typical applications of the Fly algorithm.

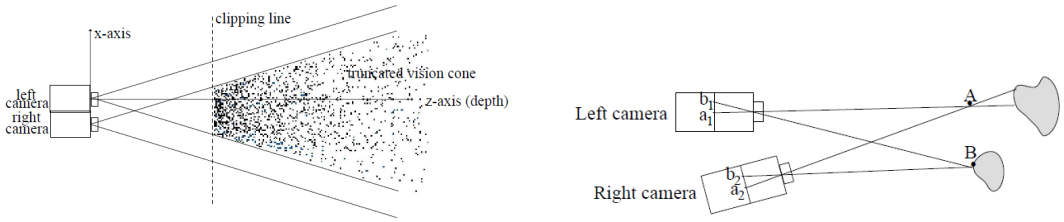
Algorithm 2 Overview of a possible implementation of a steady-state Fly algorithm

```
repeat
  Select randomly two flies f1 and f2 from the population of parents
  Compute local fitness for f1 and f2
  Select genetic operator
  if Genetic operator is new blood then
    if fitness(f1) <fitness(f2) then
      f1 ← randomly generated fly
    else
      f2 ← randomly generated fly
    end if
  else
    if Genetic operator is mutation then
      if fitness(f1) < fitness(f2) then
        f1 ← mutation(f2)
      else
        f2 ← mutation(f1)
      end if
    end if
  end if
  Compute Global fitness for all population
until algorithm is convergence
```

4.3.1 Stereo vision

A description of the stereo vision system can be found in Section 3.3.3. Here, a review is presented regarding the design of the Fly algorithm for reconstructing and detecting an object using pairs of stereo images to create a tool for mobile robot vision. The major aim of the Fly algorithm is to optimise a population of 3-D points (flies) with coordinates (x,y,z), which are randomly generated between the intersection of the two cameras' (i.e. the left and right cameras) fields of view (Figure 4.2a) [27, 103].

For illustration purposes, let us consider the 2-D case only. These coordinates are projected from both coordinates (x_L,y_L) and (x_R,y_R) of images given by the left and right cameras, respectively. The Fly algorithm was designed to detect opaque objects in a 3-D scene. Figure 4.2b demonstrates that if Fly B is on an object's surface, the projections will have similar neighbourhoods, N, in both images, i.e. the corresponding pixels in both images will usually have the same grey or colour levels. However, if Fly A is not on an object's surface, its close neighbourhoods will usually be poorly correlated because of the asymmetry of corresponding pixels and their neighbourhoods in terms



(a) Fly positions. **(b)** Camera setup with two objects.

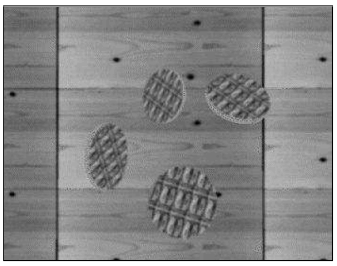
Figure 4.2: Cooperative co-evolution principles in the Fly algorithm (images from [106], courtesy of Jean Louchet).

of their illumination. This evaluation is controlled by implementing the fitness function of Equation 4.1. The fitness function measures the similarity of the corresponding projected pixels and their neighbourhoods in the two images. Thus, the highest fitness of a fly ($fitness(fly)$) means that the fly is lying on the surface of an object [26, 103, 102].

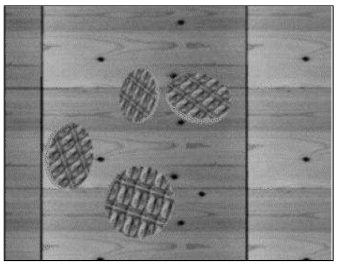
$$fitness(fly) = \frac{1}{\sum_{colours} \sum_{(i,j)\in N} [L(x_L + i, y_L + j) - R(x_R + i, y_R + j)]^2} \tag{4.1}$$

where (x_L, y_L) and (x_R, y_R) are the pixel coordinates of the left and right projections of the current individual in the pair of stereo images; $L(x_L + i, y_L + j)$ is the grey value of the left image at pixel $(x_L + i, y_L + j)$, similarly with R for the right image.

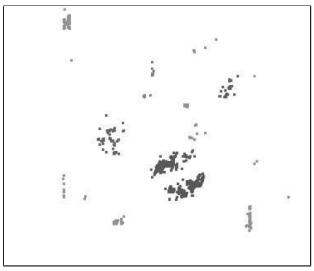
Some studies have used the standard Fly algorithm that is described above to find out the obstacles in a 3-D scene using static images, such as those in [102, 136]. To get better reconstruction results, some researchers in [126] and [128] added a simple modulation to the traditional fitness function of the Fly algorithm (turning Equation 4.1 into Equation 4.2). The latter equation includes Sobel gradient norms on the left and right projections of the fly, which is intended to penalise flies that are settled on uniform regions, i.e. the less significant flies. Sobel is a first derivative gradient filter used in image processing and in computer vision as an edge detector [67]. From an input image it outputs a new image with low values for homogeneous regions in the input image and high values for the details (such as object boundaries and noise). In [26], [27] and [106], 2-D sharing has been used to speed up algorithm convergence. Sharing prevents flies from concentrating in a small, crowded region by reducing the fitness value of the flies located in this area. Figure 4.3 demonstrates the results of applying the Fly algorithm on the synthetic “money” pair of stereo images. It shows improved results due to the



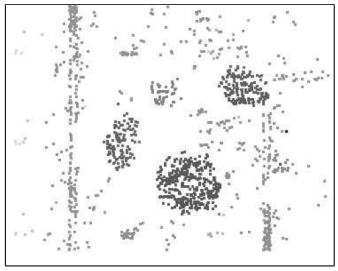
(a) "Money" image, left



(b) "Money" image, right



(c) Without sharing



(d) With sharing

Figure 4.3: Detecting objects using the Fly algorithm (images from [106], courtesy of Jean Louchet).

use of sharing (see Figure 4.3d) [106]. Furthermore, some articles have presented the idea of reconstructing an object in a 3-D scene as well as implementing stereo image sequences to innovate real-time mobile robot navigation systems [26, 102, 103, 106].

$$fitness(fly) = \frac{|\nabla(M_L)| \cdot |\nabla(M_R)|}{\sum_{colours} \sum_{(i,j)\in N} [L(x_L + i, y_L + j) - R(x_R + i, y_R + j)]^2} \tag{4.2}$$

where (x_L, y_L) and (x_R, y_R) are the coordinates of the left and right projections of the current individual, respectively; $L(x_L + i, y_L + j)$ is the grey value at the left image at pixel $(x_L + i, y_L + j)$, similarly with R for the right image; N is a neighbourhood; $|\nabla(M_L)|$ and $|\nabla(M_R)|$ are Sobel gradient norms for the left and right projections of the fly.

4.3.2 3-D tomographic reconstruction in nuclear medicine

As mentioned above, EAs have been frequently used to solve inverse problems, introducing a promising result for image processing in general and for medical imaging in particular. However, combining EAs with the tomographic reconstruction is not

trivial. A proof-of-concept has been developed in 3-D tomographic reconstruction for SPECT and PET using the Fly algorithm [28, 152, 154, 155, 156]. Each fly corresponds to a 3-D point that emits photons. An EA is then used to optimise the position of the flies. Each fly has its own image pattern. The flies cooperate to create an image pattern for the whole population. The EA will minimise the discrepancies between the input data and this image pattern. After convergence, a group of flies will correspond to the reconstructed volume. Due to the complex nature of the tomographic reconstruction problem, new genetic operators (new fitness functions, selection, mitosis, and mutations) had to be specially developed for the Fly algorithm.

In 2008, Bousquet *et al.* introduced a new method of reconstructing the 3-D radioactive concentration in nuclear medicine imaging using the Parisian (Fly) algorithm instead of the traditional expectation-maximization (EM) method. Each fly is a 3-D point in search space and serves as a photon emitter. They validated their method using real SPECT images. They initially used a relatively straightforward fitness function (a fly would receive a bonus (+1) for every of its projections that is on an object and -1 when it is not). However they discovered that small objects of low intensity disappeared with this fitness function. They addressed this problem with a new fitness function called "marginal fitness". It follows the principle of "leave-one-out cross-validation". It computes the difference between the original and two estimated images. The marginal fitness is computed as follows:

$$Local fitness(i) = \xi rror(population \setminus \{i\}) \quad - \quad \xi rror(population) \tag{4.3}$$

$\xi rror(population)$ is an error function on the image produced by the whole population. It is the distance between the total illumination pattern created by all the flies, to the actual image. It is the value that the algorithm is minimising. $\xi rror(population \setminus \{i\})$ is the same error function computed on the image produced by the population without Fly i. $Local fitness(i)$ is the marginal fitness, which is a quality measurement that should be maximised. If $Local fitness(i)$ is negative, then Fly i is a bad fly that negatively contribute to the population. If it is positive, then it is a good fly. This paper shows promising 3-D reconstruction results for both simplified synthetic data and real stenographic images in terms of its improved detection of smaller objects [28]. A generational Fly algorithm was used.

Following Bousquet *et al.* in the using marginal fitness in [28], Vidal and his colleagues (2010) posited a method of reconstructing 3-D images that takes advantage of both a steady-state Fly algorithm and PET [159]. Based on the contribution of each fly, the authors used a threshold selection in the selection stage of the algorithm instead of using classical selection operators, such as ranking, roulette wheel and tournament. That is, a fly whose fitness is positive is a good candidate for reproduction, and a fly with a negative fitness can be discarded. It makes the algorithm work faster and reduces the pressure at the selection stage. The selection process gradually eliminates flies with a negative fitness. It provides an excellent stopping criteria when the number of negative (bad) fly becomes too low. This article shows encouraging results for synthetic data.

4.4 Novelty

We can define the novelty that has been added in this thesis to the Fly algorithm for both tomography and digital arts applications as a follow:

Tomography. Our implementation focuses on aggregation part (see Figure 4.1). This include: 1) determining which flies are assigned to be in a final solution; 2) exploring a further step to exploit individual's fitness as a level of confidence; 3) voxelising the point cloud of data from a density field using implicit modelling such as Metaballs and Gaussian kernels; and 4) proposing a new mutation operator which is Directed mutation to speed-up the convergence.

Digital arts. For the first time, we apply the Fly algorithm with digital arts application. This implementation added some changes to the Fly algorithm which are: 1) changing the structure of a fly from a simple primitive (x, y, z) into a more complex structure, a vector of nine elements (x, y, z, r, g, b, α, w, h); 2) adding a texture to the Flies; and 3) investigate the benefit of a restart operator in this context.

Chapter 5

PET Reconstruction

Parts of this chapter have been exploited in two research articles published in the proceedings of the *Eurographics Workshop on Visual Computing for Biology and Medicine (VCBM)* and in the journal on *Swarm and Evolutionary Computation* respectively (see Section 1.4) [8, 7].

5.1 Introduction

This research deals with tomographic reconstruction in nuclear medicine, more particularly PET. An unknown radioactive concentration (f) is recovered from known observations ($Y = P[f]$, with P a projection operator) by solving an ill-posed inverse problem (see Figure 5.1). Y can be represented by a 'sinogram', which is the type of image that can be used as input data in tomographic reconstruction (see Figures 5.4c, 5.19 and 5.24 for examples of sinogram). The result of the reconstruction is $\hat{f}$, an estimate of f. In this chapter, we exploit the output of a CCEA, the Fly Algorithm, to improve quantitative results in reconstructed images. The Fly Algorithm evolves a population of 'flies', according to the Evolutionary Strategy paradigm. A fly is a 3-D point in the object space. Each fly is used to create projection data (the way this data is generated is problem dependant). Here the flies generate a 'simulated sinogram' ($P\left[\hat{f}\right]$, see Figure 5.3). The fitness value of each fly, i.e. a quality measurement optimised by the algorithm, is based on the consistency of its calculated projections in the images. This approach has been extended to 3-D tomographic reconstruction in medical imaging ('medical flies') [28, 152, 154, 155].

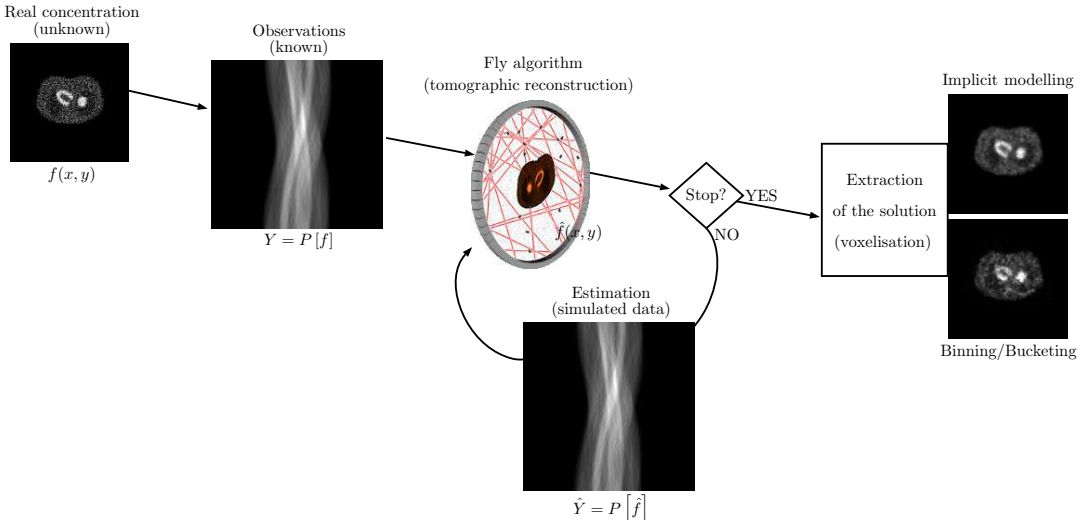


Figure 5.1: Evolutionary reconstruction using the Fly algorithm. The real radioactive concentration (f) is unknown. P is a projection operator. The projections of f are the known observations ($Y = P[f]$). Individuals of the evolutionary algorithm correspond to 3-D points. The population corresponds to an estimated radioactive concentration $\left(\hat{f}\right)$. Each individual has its own projection data. Together, they produce simulate projections $\left(\hat{Y} = P\left[\hat{f}\right]\right)$. The position of individuals is iteratively optimised using genetic operators to minimise $\mathcal{E}\left(Y, \hat{Y}\right)$ the difference between Y and $\hat{Y}$. After convergence the concentration of individuals is an estimate of the radioactive concentration.

The problem we address in this chapter is related to the extraction of the solution for PET reconstruction using the Fly algorithm (see the last main step in Figure 5.1). The chapter particularly focuses on how to voxelise and display the point cloud generated by the final fly population to produce quantatively accurate results. This step is necessary as most medical imaging software use this data representation to store and process tomographic volumes. The voxelisation problem has been overlooked in the previous work mentioned above. The 3-D space was discretised into a regular grid of volume elements (voxels). The intensity of each voxel was proportional to the number of flies located into them. To produce sharper images, only good flies were considered during the voxelisation. In this chapter, we study the collective impact of including 'marginally negative' individuals in the final solution and compare evolutionary reconstructions with those obtained using classical algorithms. The aim is to formally determine which flies are necessary in the final solution to produce a volume that is more similar to the real radioactive concentration. We also investigate and compare methods based on implicit modelling to display the fly population in order to properly minimise the difference between the reconstructed pattern ($\hat{f}$) and the ground-truth (f), and get visually realistic rendering. Our method takes advantage of the Fly Algorithm's internal data to efficiently

employ implicit modelling. In particular, we investigate how to take advantage of the individual knowledge of each fly after the evolution process to modulate their own appearance in the final image of the whole population. It is implemented using the local fitness as a level of confidence in the fly's position. The aim is to reduce the noise level and to retain edges between regions. To ascertain the validity of our new approach, evolutionary reconstructions on two numerical phantoms are compared with those obtained using two of the most popular tomographic reconstruction algorithms in nuclear medicine research, namely FBP and OSEM. FBP is only used for historical reasons. It has been supplanted in clinical routine by MLEM and its derivative OSEM. However finding suitable stopping criteria for MLEM-based methods in PET is an open research question [61]. Reconstructions improve after an unspecified number of iterations, then deteriorate.

Note that the ground-truth image f is never used during the reconstruction as it is considered unknown. However, as controlled test cases are used in this chapter, it is possible to use it for validation purposes. Qualitative validation corresponds to a visual comparison between the ground-truth image f and the corresponding reconstructions $\hat{f}$ (the estimated radioactive concentration, i.e. a volume produced using the final population of flies). In quantitative validation, the numerical difference or similarity between f and $\hat{f}$ are measured. Quantitative imaging is the aim of the research community in tomography reconstruction in nuclear medicine. In oncology, when PET scans are performed over a large period of time, quantitative imaging allows the clinician to assess the response of tumours to the treatment as it makes it possible to measure the variation in tumour volume at different stages of the disease. Quantitative validation is therefore preferred over qualitative validation in this research community. We follow this approach in this chapter.

For more background information of classical tomography reconstruction (see Section 2.4) and Parisian Evolution (see Section 4.1). The next section presents the early work on evolutionary reconstruction (see Section 5.2). Section 5.3 studies how to extract the solution of the optimisation problem in this co-operative co-evolution scheme, i.e. which individuals should be considered during the voxelisation. Section 5.4 focuses on a technique called Implicit Modelling. An overview is given in Section 5.4.1. Section 5.4.2 shows how to apply this technique during the voxelisation and Section 5.4.3

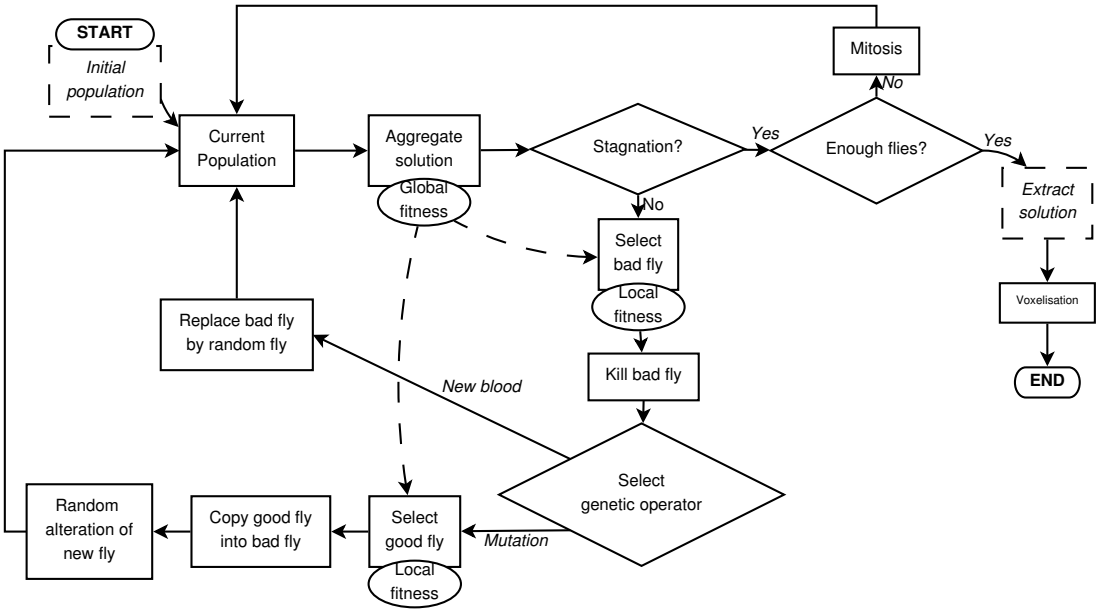


Figure 5.2: The Fly algorithm is an iterative method (as described in Figure 2.4). Here genetic operators (new blood and mutations) are applied to correct the position of flies and minimise the error between the known observations ($P[f]$) and the projection data ($P\left[\hat{f}\right]$) generated by the population of flies ($\hat{f}$). After convergence the concentration of flies is an estimate of the radioactive concentration.

shows how to exploit the marginal fitness of each fly during the final voxelisation to produce high fidelity reconstructed volumes. In Section 5.5, we analyse the results obtained using controlled test cases of increasing complexity. The final section provides a discussion and concluding remarks.

5.2 Early evolutionary reconstruction

The Fly algorithm has been extended more recently to 3-D reconstruction in medical imaging ('medical flies') [28, 152, 154, 155]. Figure 5.2 illustrates how the Fly algorithm works in this case. Here, each fly emulates a radioactive emitter and has its own illumination pattern. The projection data they create can be stored as a sinogram. It is 2-D image made of a set of 1-D projections at successive angles. Figure 5.3 shows how the flies are orthogonally projected to generate parts of a sinogram. The data created by all the flies $\left(\hat{f}\right)$ is aggregated to build the sinogram simulated by the population $\left(P\left[\hat{f}\right]\right)$. It is used to compute a metrics useful in the calculation of the fitness functions. At the start of the reconstruction, flies are randomly scattered in the search space. The evolutionary algorithm optimises their positions to minimise the

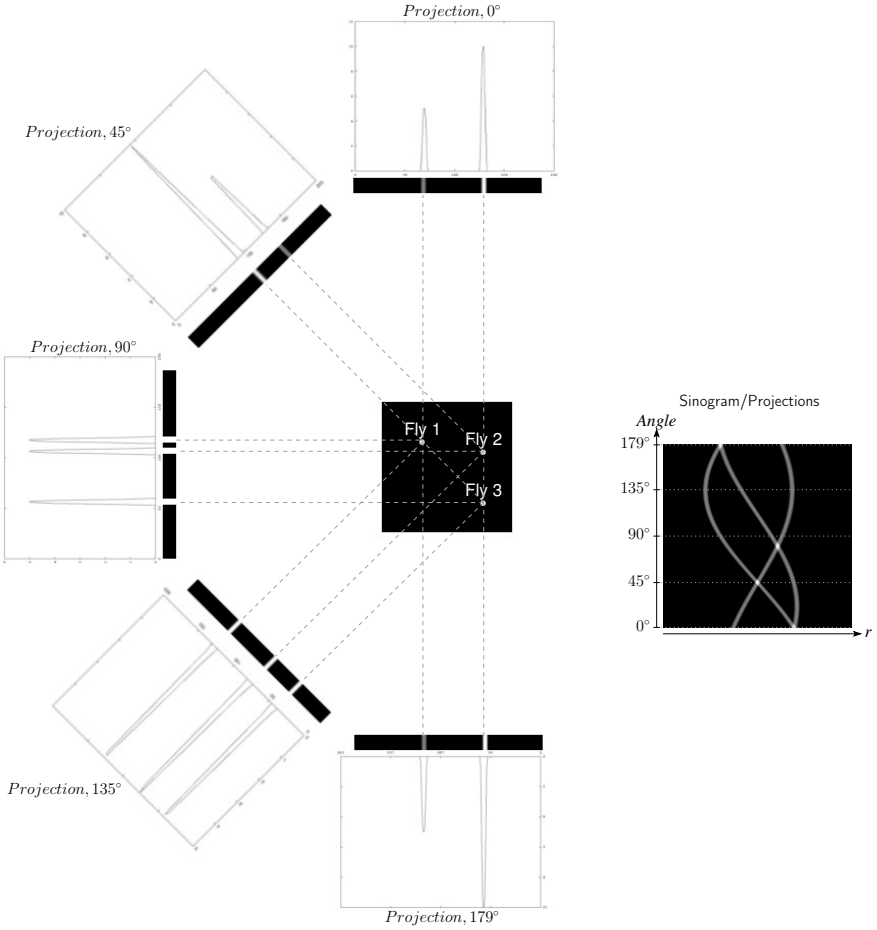


Figure 5.3: From a population of flies ($\hat{f}$) to an estimated sinogram $\left(P\left[\hat{f}\right]\right)$. Each fly has its own projection data at different angles. Put together, they produce the estimated sinogram.

global fitness function. After convergence, $P\left[\hat{f}\right]$ (the sum of illumination patterns of all the flies) closely matches the input data ($P[f]$) and the final population of flies ($\hat{f}$) gives an estimate of the unknown radioactive concentration (f).

To optimise the flies' position, our algorithm relies on mutation and new blood operators. Again, no cross-over is used as it is not particularly useful in the Fly algorithm for image reconstruction. Let us consider two flies that are well positioned. If there are in two separate areas, then it does not make sense to create a new fly in between as it would probably lead to a bad fly. The new blood operator aims at maintaining some diversity in the population. This is particularly important at the early stages of the reconstruction to make sure no small object of low intensity are missed. Its principle is relatively simple:

1. A bad fly is randomly selected using our Threshold selection (see below for an explanation)

2. Its projections are removed from $P\left[\hat{f}\right]$

3. The fly is replaced by a new fly randomly positioned in the search space, and

4. The projections of the new fly are added to $P\left[\hat{f}\right]$.

See the next chapter for more information on mutation.

To evaluate the performance of successive populations toward more realistic images, we use a distance measurement ($\mathcal{E}$) between global images $\left(P\left[\hat{f}\right]\right)$ resulting from the photon emissions of the population of flies $\left(\hat{f}\right)$ and the real images ($Y = P[f]$) obtained from the sensors. It is the global fitness of the population:

$$fitness = \mathcal{E}\left(\hat{f}\right) = \frac{1}{w \times h}\left\|Y - P\left[\hat{f}\right]\right\|_2^2 \tag{5.1}$$

with $\mathcal{E}$ based on the ℓ^2-norm (also known as the Euclidean distance) between the observations (Y) and the data simulated $\left(P\left[\hat{f}\right]\right)$ by the flies; w and h the number of pixels in Y and $P\left[\hat{f}\right]$ along the horizontal and vertical axis respectively; $\hat{f}$ the fly population (i.e. an estimate of the unknown f), which corresponds to the population of flies; P is the projection operator, which projects flies to simulate an estimate of Y. The algorithm minimises the fitness as follows:

$$\hat{f} = \underset{f \in \mathbb{R}^2}{\arg\min}\left(\frac{1}{w \times h}\left\|Y - P\left[\hat{f}\right]\right\|_2^2\right) \tag{5.2}$$

$\mathcal{E}$ measures the discrepancies between the observations and the data simulated from flies. Lower values of $\mathcal{E}$ correspond to lower errors in the data simulated by the flies, i.e. an image of the population $\hat{f}$ that better matches the observations.

The fitness of each individual fly is calculated as its (positive or negative) contribution to the collective fitness of the population (which is called 'marginal fitness evaluation') [28]. It is based on the "leave-one-out-cross-validation" principle. In practice, we measure the population's performance twice: once taking into account all the individuals (i.e. the global fitness); and once without the fly (i) that is being assessed. The two values are

then compared. This way, we know if a fly helps the population improve its performance or not:

$$F_m(i) = \mathcal{E}\left(\hat{f} \setminus \{i\}\right) - \mathcal{E}\left(\hat{f}\right) \tag{5.3}$$

where: $F_m(i)$ the marginal fitness of Fly i, $\left(\hat{f} \setminus \{i\}\right)$ is the population without Fly i. The marginal fitness makes it possible to detect if a fly is positively or negatively contributing to the population's performance:

- If the local fitness is positive, then the fly improves the population;
- If it is negative, then the fly reduces the population's performance;
- If it is null, then the fly has no impact on the population's performance.

F_m is a measure that is maximised by the algorithm. Note that due to the cooperation between individuals, the Fly algorithm is able to minimise the global fitness. It is computed in F_m only. During the evolution process, the number of flies whose fitness is negative decreases; and the number of flies whose fitness is positive increases. The Threshold Selection will get stuck when the algorithm converges ('stagnation'). This provides an excellent stopping criterion.

We are using this feature to introduce progressive multi-resolution processing [155]. The evolution starts with a relatively low number of flies. Each time stagnation is detected, evolution is paused and a mitosis process is launched. Similar to biological mitosis, for each fly a new fly is created by mutation and added to the population: This doubles the population size. Evolution eventually resumes after the mitosis. The whole evolution-mitosis process is stopped when after two successive mitosis the global fitness does not improve anymore.

Once the optimisation loop ends, the solution has to be extracted and encoded. Contrary to mainstream tomographic reconstruction algorithms whose output is a 3-D rectilinear grid of voxels, our Fly-based approach delivers a set of 3-D points as an output. Deciding which type of representation is more legible to the user is another story. Authors in [143] shows the advantages of a representation based on discrete 3-D points. In order to enable a genuine (/trusty) comparison between the outputs of the Fly algorithms and

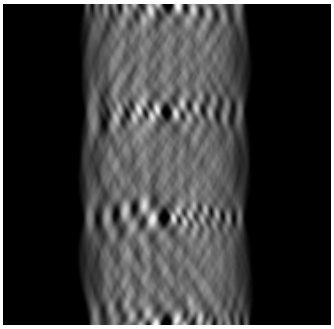
mainstream methods, we show in this research how to do the opposite way and build a continuous representation of the Fly output. It also makes it possible to use a multitude of image processing and visualisation tools developed for voxelised data.

To date, only the sub-population of flies with a positive fitness $\left(\hat{f}^{+}\right)$ was taken into account to build the final estimate of the distribution of 3-D points [154]. It is sampled into voxels. Data binning, also called bucketing, has been used so far to produce the voxel map. The 3-D space is divided into a regular grid. Each element of the grid is called a voxel. With data binning, the value of a voxel is given by the number of flies that it contains. In this chapter, we study which flies have to be included in the final result to improve accuracy, and how to best voxelise the fly data using implicit modelling to further improve quantitative results.

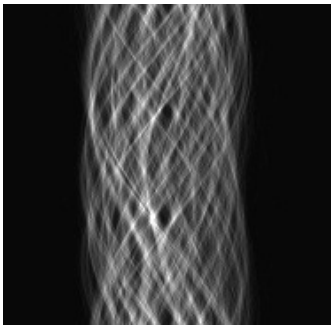
5.3 Extraction of the solution

Traditionally, the answer to an optimisation problem modelled using artificial evolution is the best individual of the whole population after convergence. Using the cooperative co-evolution scheme of the Fly algorithm, the solution of the optimisation problem is embedded within the population [102]. We use tomographic reconstruction as an application example of the Fly algorithm but other applications could be considered. As a proof-of-concept, below we will consider the 2-D case only. However, note that the algorithm is actually developed for 3-D and the notations can be extended to account for the Z-dimension.

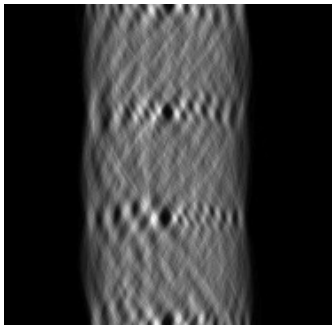
Figure 5.4a shows the input data. It is the known observations stored as a sinogram. The projection operator P is designed to project the population of flies $\left(\hat{f}\right)$ in order to simulate the sinogram $\left(P\left[\hat{f}\right]\right)$ according to the illustration in Figure 5.3. Figure 5.4c shows the corresponding simulated sinogram after convergence of 12,800 flies. Figure 5.4b shows the sinogram generated by flies $\left(\hat{f}^{+}\right)$ with a positive fitness only. We can observe that the sinogram produced by all the flies is visually closer to the ground-truth than the sinogram simulated by good flies only.



(a) Known observations ($Y = P[f]$).



(b) Estimation with the good flies only $\left(P\left[\hat{f}^{+}\right]\right)$ (NCC with (a): 90.97%).



(c) Estimation with all the flies $\left(P\left[\hat{f}\right]\right)$ (NCC with (a): 99.19%).

Figure 5.4: Sinograms with 185 pixels per projection, 1st angle: 0°, angular step: 1°, and last angle: 179°. Corresponding radioactive concentrations are given in Figure 5.5. The geometrical relationship link between the simulated sinogram $\left(P\left[\hat{f}\right]\right)$ from the position of flies ($\hat{f}$) is given in Figure 5.3.

To measure the level of similarity between two images I_1 and I_2 of $m \times n$ pixels, we use the normalised cross-correlation (NCC):

$$NCC(I_1, I_2) = \frac{1}{m \times n} \sum_{i=0}^{i<m} \sum_{j=0}^{j<n} \frac{\left(I_1(i,j) - \bar{I}_1\right)\left(I_2(i,j) - \bar{I}_2\right)}{\sigma_1 \sigma_2} \tag{5.4}$$

with $\bar{I}_1$ and $\bar{I}_2$ the average values of all the pixels in I_1 and I_2 respectively, such as

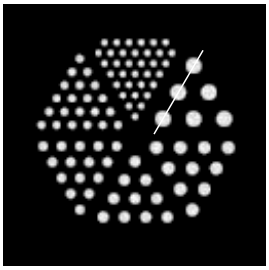
$$\bar{I} = \frac{1}{m \times n} \sum_{i=0}^{i<m} \sum_{j=0}^{j<n} |I(i,j)| \tag{5.5}$$

and σ_1 and σ_2 the standard deviations of all the pixel values in I_1 and I_2 respectively, such as:

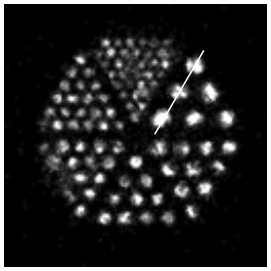
$$\sigma = \sqrt{\frac{1}{m \times n} \sum_{i=0}^{i<m} \sum_{j=0}^{j<n} \left[I(i,j) - \bar{I}\right]^2} \tag{5.6}$$

Due to the stochastic nature of the algorithm, 15 evolutionary reconstructions have been performed in total to provide statistically meaningful results. The image simulated using both good and bad flies leads to a NCC of 99.27% ±0.06%. The image simulated by the good flies only has a NCC of 91.40% ±0.75%. In other words, keeping the flies with a negative fitness leads to more accurate and more stable results.

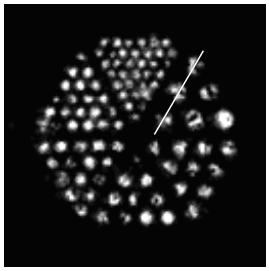
The concentration of flies is then sampled into voxels to generate the tomography volume. For data binning, we considered a fly as a Dirac delta function (δ): The value of each



(a) Ground-truth (f). It is unknown.



(b) Concentration estimation with all the flies $\left(\hat{f}\right)$ (NCC with (a): 82.74%).



(c) Concentration estimation with the good flies only $\left(\hat{f}^{+}\right)$ (NCC with (a): 82.24%).

Figure 5.5: Tomographic reconstruction using 12,800 flies. Corresponding sinograms are given in Figure 5.4. The geometrical relationship link between the position of flies ($\hat{f}$) and the simulated sinogram ($P\left[\hat{f}\right]$) is illustrated in Figure 5.3.

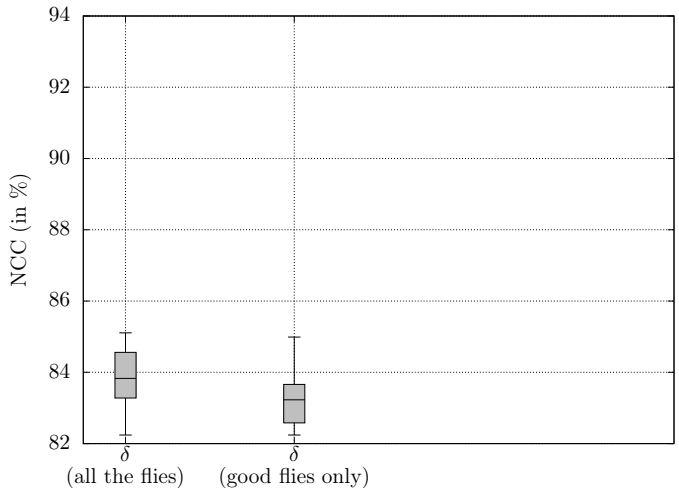


Figure 5.6: Similarity metrics (NCC) between the ground-truth (f) and the images of the fly population ($\hat{f}$ and $\hat{f}^{+}$) using the binning method for voxelisation. Due to the stochastic nature of the evolutionary reconstruction, the reconstruction is performed 15 times to produce statistically meaningful results.

voxel is incremented for each fly it contains. Figure 5.5 shows the ground-truth image and the corresponding reconstructions (qualitative or visual validation). Figure 5.6 presents the NCC between the ground-truth and the reconstructed images (quantitative validation). Both figures complement each other and show that reconstructions including flies with negative fitness generally produce images that are visually and numerically closer to the ground-truth. In past papers, we usually kept good flies only as it resulted into visually sharper reconstructed images.

In order to measure how sharp images are, we compute the sum of gradient magnitudes for each reconstruction:

$$Sharpness(I) = \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} |\nabla I|\,(i,j) \tag{5.7}$$

It is 4120 ±320 with good flies only; 4088 ±497 with all the flies. Removing negative flies leads to sharper reconstructions. Note that we use other metrics below to ascertain this assumption as Eq. 5.7 is sensitive to noise.

Removing all the flies with a negative fitness is not appropriate as the Fly algorithm is based on a co-operative scheme. When a fly is killed, i) its contribution to the population is removed, ii) the global fitness changes, iii) which also modifies the local fitness of every other fly. In other words, when any fly is killed, a good fly may become bad, and *vice versa*, a bad fly may become good. Because of this phenomenon, we can eventually say that bad flies have to be included in the final solution. This is why the NCC of the whole population (including bad and good flies) is better on average than the sub-population of good flies only.

Figure 5.7 corresponds to profiles (also known as intensity profiles) extracted from white lines in Figure 5.5. In the imaging context, a profile corresponds to a set of intensity values taken from regularly spaced points along an arbitrary line segment within the image. It is often plotted as a 1-D function. We quantify how steep edges are using the rise time from 10% to 90% and fall time from 90% to 10%. They are useful metrics used in electronics and signal processing to quantify the time taken by an approximation of a step function to change from a specified extremum to another specified extremum [9]. In our context, we can consider the profiles as a succession of step functions. Here these metrics assess how many pixels are required to change from the minimum (background: no radioactivity) to maximum (hot rodes: radioactivity) and *vice versa*. In the ideal case, they are perfect step functions, and the rise time and fall time should be close to 1 pixel. In real application such as PET reconstruction, there is blur: the values are likely to be greater than 1. There is also noise in real data, which could add a bias in the numerical value. To account for noise, only the samples within 10% to 90% of the whole interval of the signal are considered. The values for each edge of every profile are summarised in Table 5.1. Edges are usually steeper

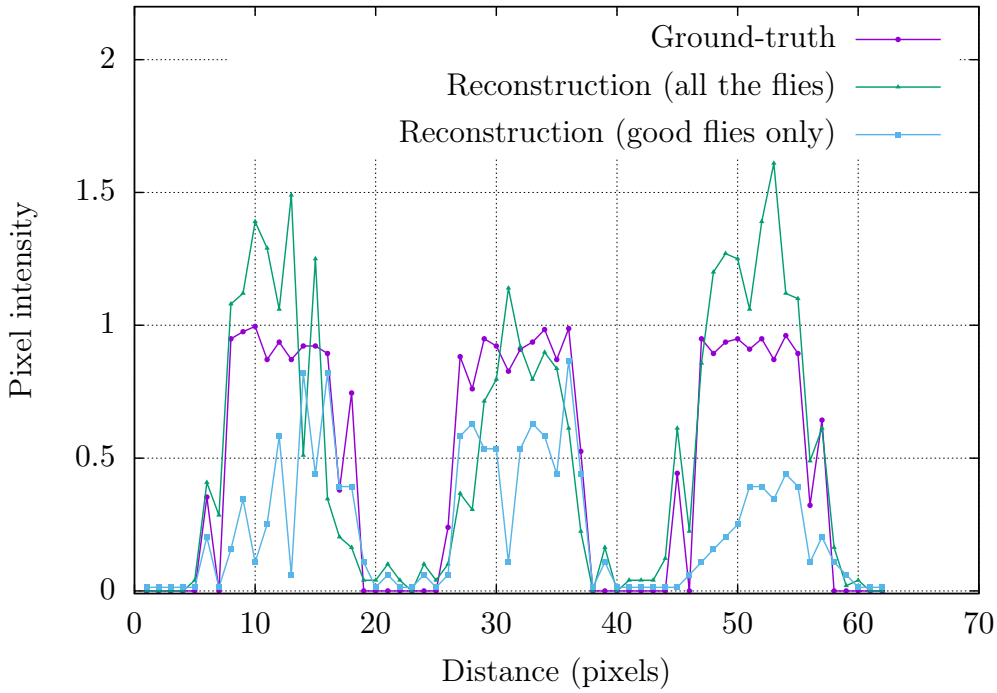


Figure 5.7: Intensity profiles corresponding to the white lines in Figures 5.5a, 5.5b and 5.5c.

Table 5.1: Image and profile comparison between the ground-truth (Figure 5.5a) and the evolutionary reconstructions (Figures 5.5b, 5.5c, 5.12, and 5.15). Numerical values in bold characters are the ones closest to the ground-truth.

	NCC	Sharpness	Rise time	Fall time	∑ Profile
Ground-truth	N/A%	9443	2.22	2.35	29.33
All the flies	82.74%	3857	4.17	3.44	32.13
Good flies only	82.24%	3931	**3.15**	**3.08**	14.34
Metaballs	89.69%	5150	3.90	3.90	31.96
Gaussian	**92.79%**	**6303**	3.57	3.36	**31.63**

for the ground-truth and the reconstruction using the good flies only. The transition from extreme values is twice as large for the reconstruction with all the flies than the ground-truth.

In addition we check the total sum of values of each profile. It is: 29.33 for the ground-truth; 29.09 ±4.34 for all the flies; 21.88 ±4.34 for the good flies only. It indicates that there is approximately as much information in the profiles of the ground-truth and all the flies, and some information is missing in the profile of the good flies only.

The results above can be summarised as follows:

1. Edges are much more blurred when bad flies are included.

2. Gaps appear when bad flies are not included.

It could be seen as a dilemma,

- Bad flies should be excluded to preserve edges;
- Bad flies should be included to avoid holes in the data.

In the next section of this chapter, we demonstrate how to solve this dilemma: Including bad flies to produce a more accurate image, whilst still retaining its sharpness.

5.4 Voxelisation using implicit modelling

We saw in the previous section that the solution that is extracted should contain all the flies, including the ones with a negative marginal fitness. Previously the contribution of each fly in the final volume was one and it was assigned to a single voxel. It could lead to noise. It is not uncommon to post-process tomographic images with a low-pass convolution filter. However, with our pixel/voxel-less approach it is possible to use the internal data of the Fly algorithm, here the flies' position, to remove the need for a smoothing filter. Below we demonstrate how to spread this contribution over several voxels using implicit modelling.

5.4.1 Definition

Implicit modelling is a CG technique used to define the surface of geometric objects using control primitives (e.g. points or line segments) and a few equations [23]. Blobby Molecules, Metaballs and Soft Objects are well known types of implicit modelling techniques. In computer graphics, it consists of the steps below:

1. Positioning control primitives (usually points or line segments) in the 3-D space;

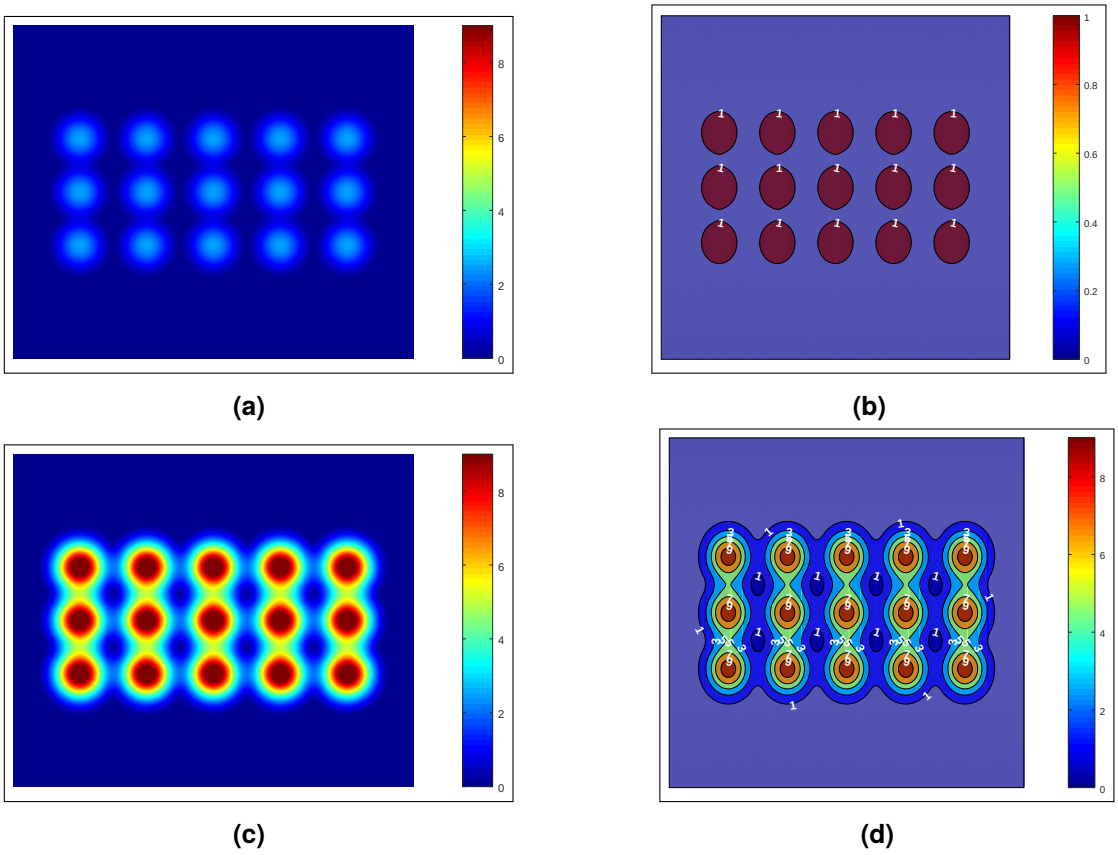


Figure 5.8: 3-D density field using 15 points as control primitives using Eq. 5.9. When the particles are in close proximity, their density fields are joining each other smoothly without discontinuity. (a) and (c): cross-sections of the density field at different heights. (b) and (d) corresponding isolines. See Figure 5.9 for corresponding 3-D isosurfaces.

2. Computing the corresponding density field using a given equation (e.g. Eqs. 5.8 or 5.9) (see Figures 5.8a and 5.8c);

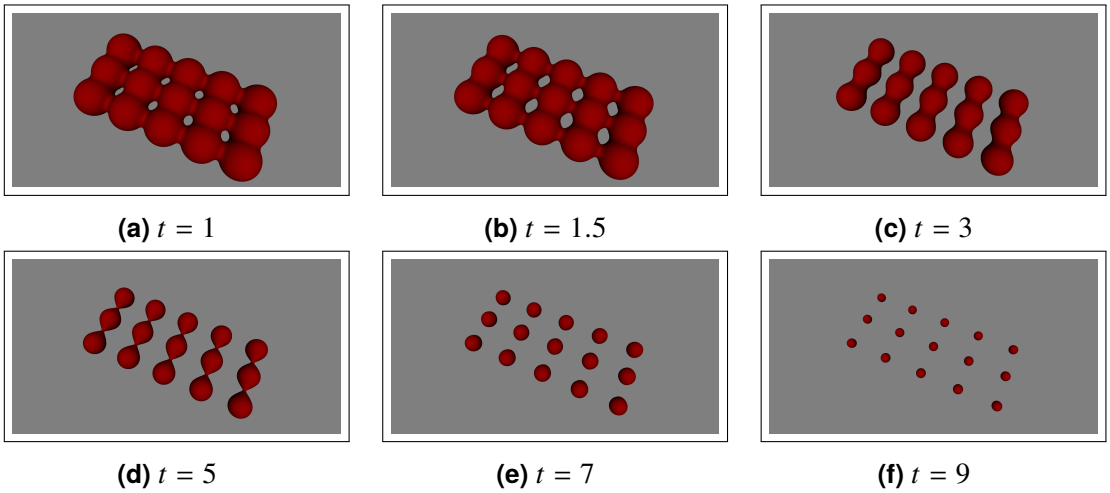
3. Selecting a threshold value (see Figures 5.8b and 5.8d);

4. Reconstructing the isosurface corresponding to the threshold using either raycasting [95] or marching cubes [101] (see Figure 5.9).

Blobby Molecules [23] uses the electron density distribution of the hydrogen atoms (Gaussian Distribution):

$$f(r) = ae^{-br^2} \tag{5.8}$$

b is related to the standard deviation of a Gaussian curve, a is the height of the curve, and r is the distance to the atom centre (see Figure 5.10).



(a) $t = 1$ **(b)** $t = 1.5$ **(c)** $t = 3$

(d) $t = 5$ **(e)** $t = 7$ **(f)** $t = 9$

Figure 5.9: Implicit surfaces corresponding to the density field defined with the 15 metaballs of Figure 5.8. Triangle meshes are extracted from the density field using the Marching Cubes algorithm [101] with various threshold (t) values.

For Metaballs [118], the density field is modelled using a piecewise function:

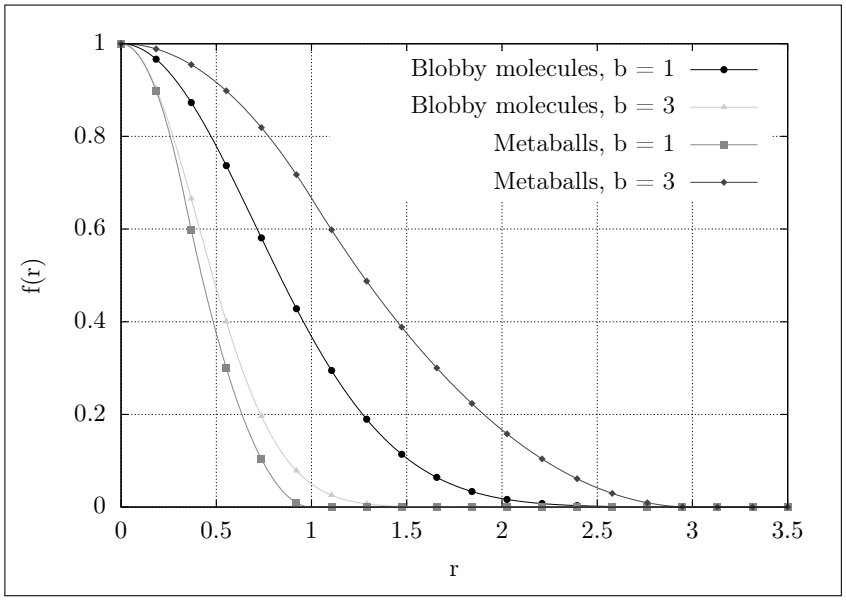
$$f(r) = \begin{cases} a\left(1 - \frac{3r^2}{b^2}\right) & \forall r \in [0; b/3] \\ \frac{3a}{2}\left(1 - \frac{r}{b}\right)^2 & \forall r \in [b/3; b] \\ 0 & otherwise \end{cases} \tag{5.9}$$

Figure 5.11 illustrates how the three sub-functions from Eq. 5.9 are combined to produce a smooth falling curve. The influence of the parameters a and b are presented in Fig 5.10. a is a scaling factor, and b is the maximum distance that a control primitive contributes to the field.

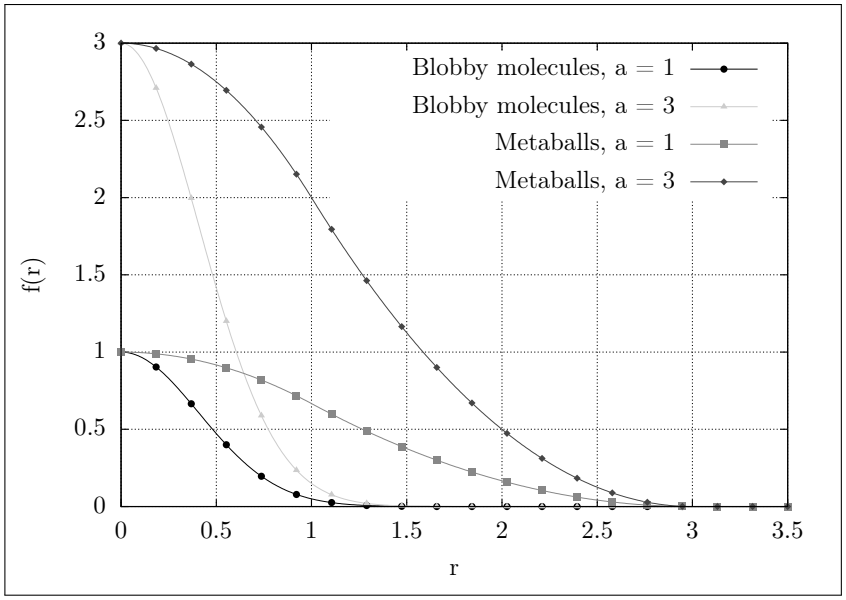
The value of the density field at a any point $[x, y, z]$ is given by:

$$F(x, y, z) = \sum_{i=1}^{N} f(\sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}) \tag{5.10}$$

with N the number of control points and $[x_i, y_i, z_i]$ the position of the i-th control point. When two particles are close to each other, their density fields are merging in a smooth manner (see Figure 5.8c). When they are sufficiently far apart, their density fields stay separated. Evaluating $F(x, y, z)$ becomes computationally expensive when the number of control primitives increases. To limit this effect, the field function in Eq. 5.9 does not make use of the exponential and it is bounded as $f(r)$ does not contribute much to



(a) With $a = 1$.



(b) With $b = 3$.

Figure 5.10: Density field control functions from Eqs. 5.8 and 5.9. Parameters a and b are used to control the height and the width of the curve. For a given value of b, the shape of the curve can be more or less wide depending on the density field function used.

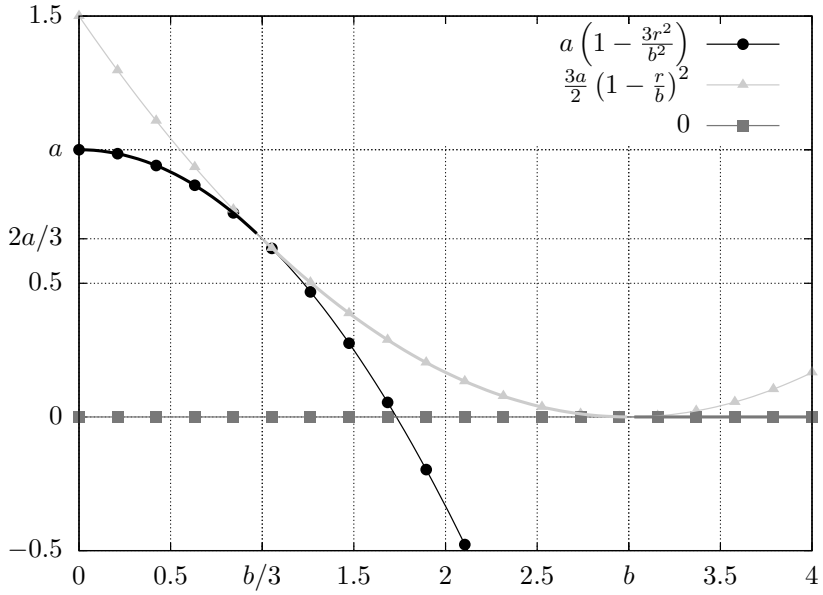


Figure 5.11: Decomposition of $f(r)$ from Eq. 5.9 with $a = 1$ and $b = 3$. Eq. 5.9 is a piecewise function with 3 sub-functions that join each other in $b/3$ and b to produce a smooth falling curve.

the field when r increases. It makes density fields using Metaballs faster to compute than those with Blobby molecules.

Other field functions, such as Soft Objects [169], are of course possible but will not be investigated in this research. We focus here on Blobby Molecules as they rely on Gaussian kernels; and on Metaballs as they are well known in the CG community.

5.4.2 Voxelisation using Metaball as density field function

The stochastic nature of the evolutionary algorithm leads to noisy PET volumes (see Figures 5.5 and 5.7). To limit noise, i) voxelised volume could be post-processed by a low-pass filter, it would lead to a loss of information, or ii) more flies can be used in the reconstruction, the computing time will significantly increase.

The aim of the Fly algorithm is to estimate the radioactive concentration. As an output it produces a 'point cloud'. This point cloud can be described as a density field. Instead of the δ function, an implicit function ($f(r)$) is used: Here, a fly corresponds to a particle surrounded by a density field. Using Equations 5.9 and 5.10, the influence of the particle decreases with the distance from the particle location.

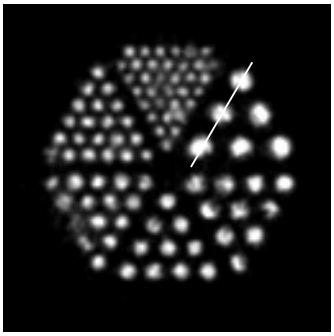


Figure 5.12: Voxelisation of the fly population ($\hat{f}$) using 12,800 metaballs (NCC with ground-truth: 89.69%) (see Figure 5.5a for the corresponding ground-truth). The same fly population as in Figure 5.5b was used.

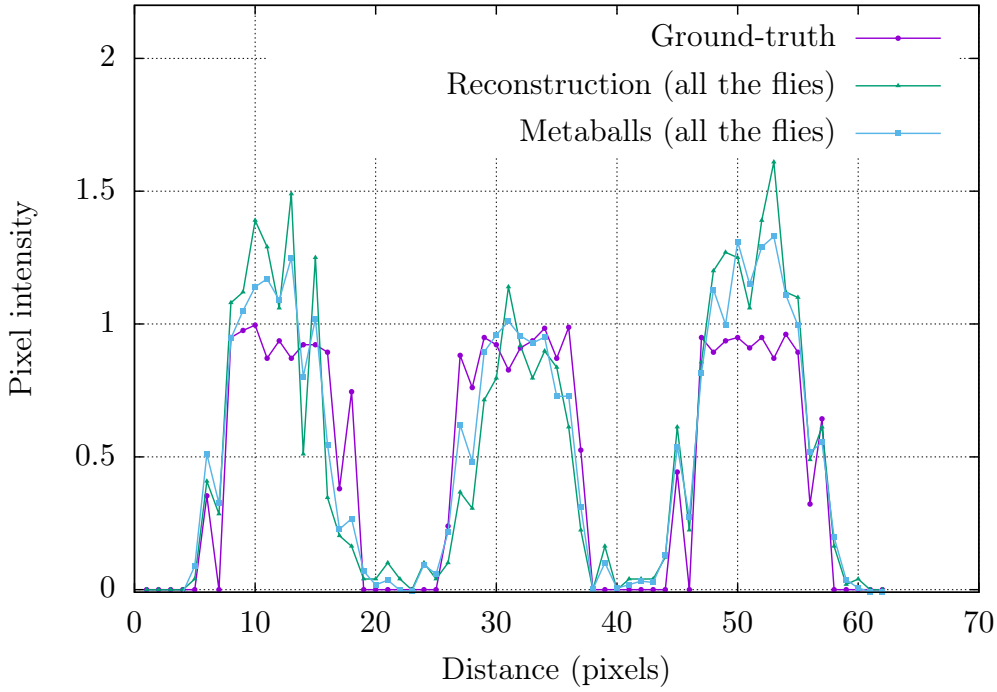


Figure 5.13: Intensity profiles corresponding to the white lines in Figures 5.5a, 5.5b and 5.12.

Figure 5.12 shows the reconstruction results when Metaballs are used. The NCC with the ground-truth is 89.69% in this case. Profiles corresponding to the ground-truth, all the flies, and the Metaballs are shown in Figure 5.13. The total sum of values of the profile for metaballs is 31.76 ±3.44, which is relatively close to the corresponding value in the ground-truth (29.33). The average rise time and fall time between 10% and 90% are both 3.9 pixels. It indicates that the sharpness around large objects is still not well recovered. It is because a fly is spread over several voxels. As a consequence, flies leak at edges, which leads to unsharpness. However, the overall sharpness metrics (see Eq. 5.7) is 4964 ±429. This is because smaller structures are recovered. In Figure 5.14, we can see that the reconstructions are consistent during 15 different runs. The NCC

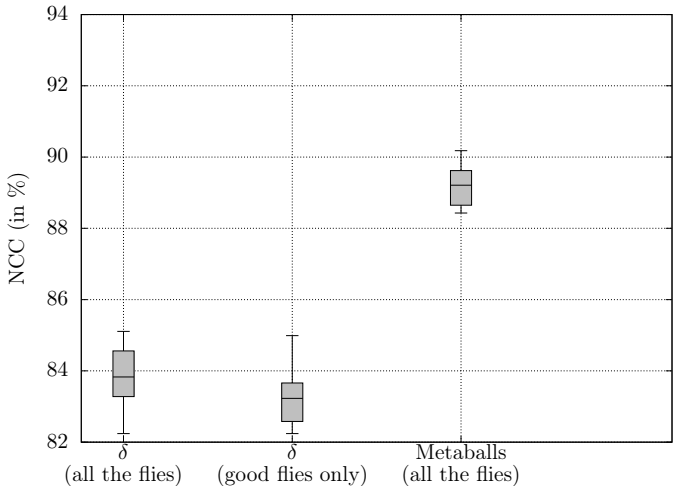


Figure 5.14: Similarity metrics (NCC) between the ground-truth (f) and the images of the fly population using the binning method and Metaballs for voxelisation.

with 12,800 flies to generate a density field is much better than using our previous voxelisation method based on binning.

5.4.3 Adaptive Gaussian kernels to exploit the fly's individual knowledge

Although the images produced using Metaballs are quantitatively much better than using a naïve approach based on binning (e.g. noise reduction), the final output could be better if edges between areas of different concentrations could be preserved. In this section, we will use Gaussian kernels instead of Metaballs. For each fly, the spread of the Gaussian function will depend on the fly's marginal fitness. This is because we can consider this numerical value as a level of confidence in the fly's position. We chose Gaussian kernels as the Gaussian function is very well known and it is used for Blobby Molecules [23] (see Eq. 5.8).

The total contribution of every fly to the final volume is 1. When we were using (δ), each fly was embedded into one voxel only. Using metaballs, it was spread over several voxels. In this case it is not straightforward to normalise the fly's contribution to account for its performance by modulating a and b in Eq. 5.9 depending on F_m. It would require to compute for each fly:

$$\int_0^b f(r)\,\mathrm{d}r \;\forall\; a \,\&\, b \tag{5.11}$$

As an alternative, we consider each fly as a Gaussian kernel whose standard deviation is linearly proportional to F_m. The greater F_m, the more accurate the fly position. The lower F_m, the lower the trust in the fly position.

$f_i(r)$ in Eq. 5.10 becomes:

$$f_i(x, y, z, \sigma_i) = \frac{1}{V_i} \exp\left(-\left(\frac{(x - x_i)^2}{2\sigma_i^2} + \frac{(y - y_i)^2}{2\sigma_i^2} + \frac{(z - z_i)^2}{2\sigma_i^2}\right)\right) \tag{5.12}$$

with

$$V_i = 2\pi\sigma_i^3 \times \sqrt{2\pi} \tag{5.13}$$

and

$$\sigma_i = \sigma_m + (\sigma_M - \sigma_m) \times \left(1 - \frac{F_m(i) - \min(F_m)}{\max(F_m) - \min(F_m)}\right) \tag{5.14}$$

where (x, y, z) is the position of the voxel in the object space, (x_i, y_i, z_i) is the position of Fly i in the object space, σ_i is the standard deviation for Fly i, σ_m and σ_M are the lowest and largest possible standard deviations, and $\min(F_m)$ and $\max(F_m)$ the smallest and biggest marginal fitness of flies. V_i is used to ensure that the total contribution of each fly to the final volume is 1.

We can see the corresponding reconstruction in Figure 5.15. In this case the NCC with the ground-truth is 92.79%. It looks visually closer to the ground-truth than any of the previous reconstructions. The average rise time and fall time in the profile are 3.57 and 3.36 pixels respectively (see Figure 5.16). The overall sharpness metrics is 5710 ±309, which is better than any of the previous values. The total sum of values of the profile

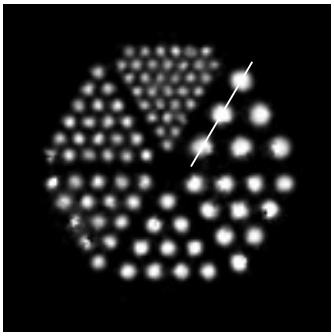


Figure 5.15: Voxelisation of the fly population ($\hat{f}$) using 12,800 gaussian kernels (NCC with ground-truth: 92.79%) (see Figure 5.5a for the corresponding ground-truth). The same fly population as in Figures 5.5b and 5.12 was used.

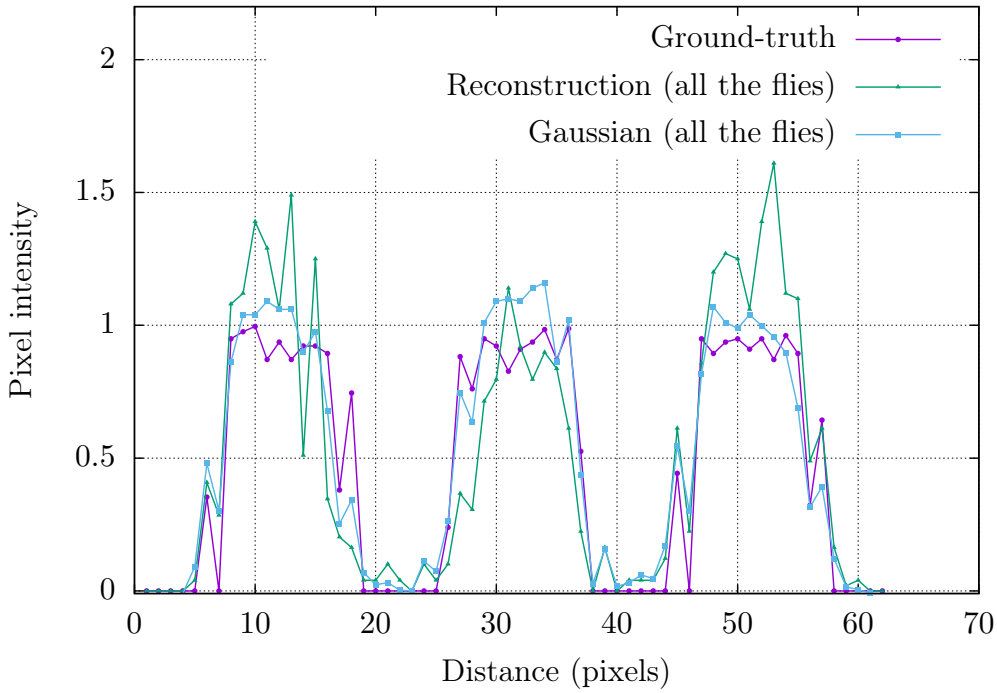


Figure 5.16: Intensity profiles corresponding to the white lines in Figures 5.5a, 5.5b and 5.15.

for Gaussian kernels is 32.18 ±2.36, which is also close to the corresponding value in the ground-truth. Figure 5.17 shows that the NCC for 15 reconstructions between the new reconstruction and the ground-truth is further improved. Results are consistent and the NCC is 10% higher than our previous method based on binning.

One of our main aims in this chapter was to demonstrate that more sophisticated voxelisation in the Fly algorithm could lead to better reconstructions. In the test-case considered so far, the final reconstruction is closer to the ground-truth and edges have been preserved. The fly population is now considered as a density field. The spread of Gaussian kernels is individually modulated depending on the marginal fitness of each fly. The outcome is a resolution-free model that is truly scalable, e.g. for a given reconstruction the noise level does not increase with the number of voxels.

5.5 Evaluation and comparative study

In this section, we evaluate our method using controlled test-case of increasing complexity. We also compare our reconstructions with those obtained with FBP and OSEM. The first step is to compare our various voxelisation methods in term of speed

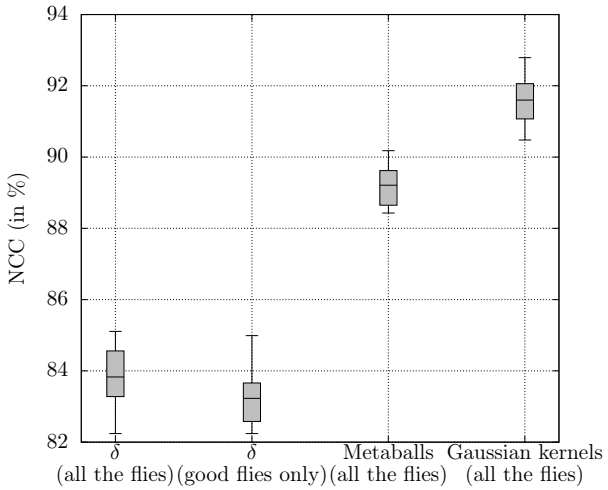


Figure 5.17: Similarity metrics (NCC) between the ground-truth (f) and the images of the fly population using the binning method, Metaballs and Gaussian kernels for voxelisation.

and accuracy (see Section 5.5.1). The second step is to reconstruct volumes using a degraded sinogram from the same phantom to increase the problem's complexity (see Section 5.5.2). Finally, we try the reconstruction methods on a more anatomically realistic phantom (see Section 5.5.3).

5.5.1 Hot rode phantoms (ideal case)

Figure 5.18 is an illustration of the reconstructions obtained at different stages of the algorithm with the sampling techniques presented above. It can be seen that using the local fitness to adapt the width and height of Gaussian kernels for each fly provide the most visually realistic results and also the most accurate results in term of NCC. Computations were performed on HPC Wales' supercomputer using a single node with an Intel Xeon Westmere X5650 @ 2.67 GHz processor [79, 82]. A quick succession of mitosis happens in less than 5 minutes, when the NCC reaches a threshold. Optimal results were obtained in 8:41 minutes using 6,400 flies (NCC of 92.76%). More processing time did not yield to better results. In fact, using 12,800 flies (17:57 minutes) leads to results that are comparable to using 3,200 flies (4:50 minutes): NCC of 91.67% and 91.33% respectively. Also, it shows the benefit of implicit modelling over binning. About 18 minutes are required to reach a NCC of 80% with binning, only 2 with implicit modelling.

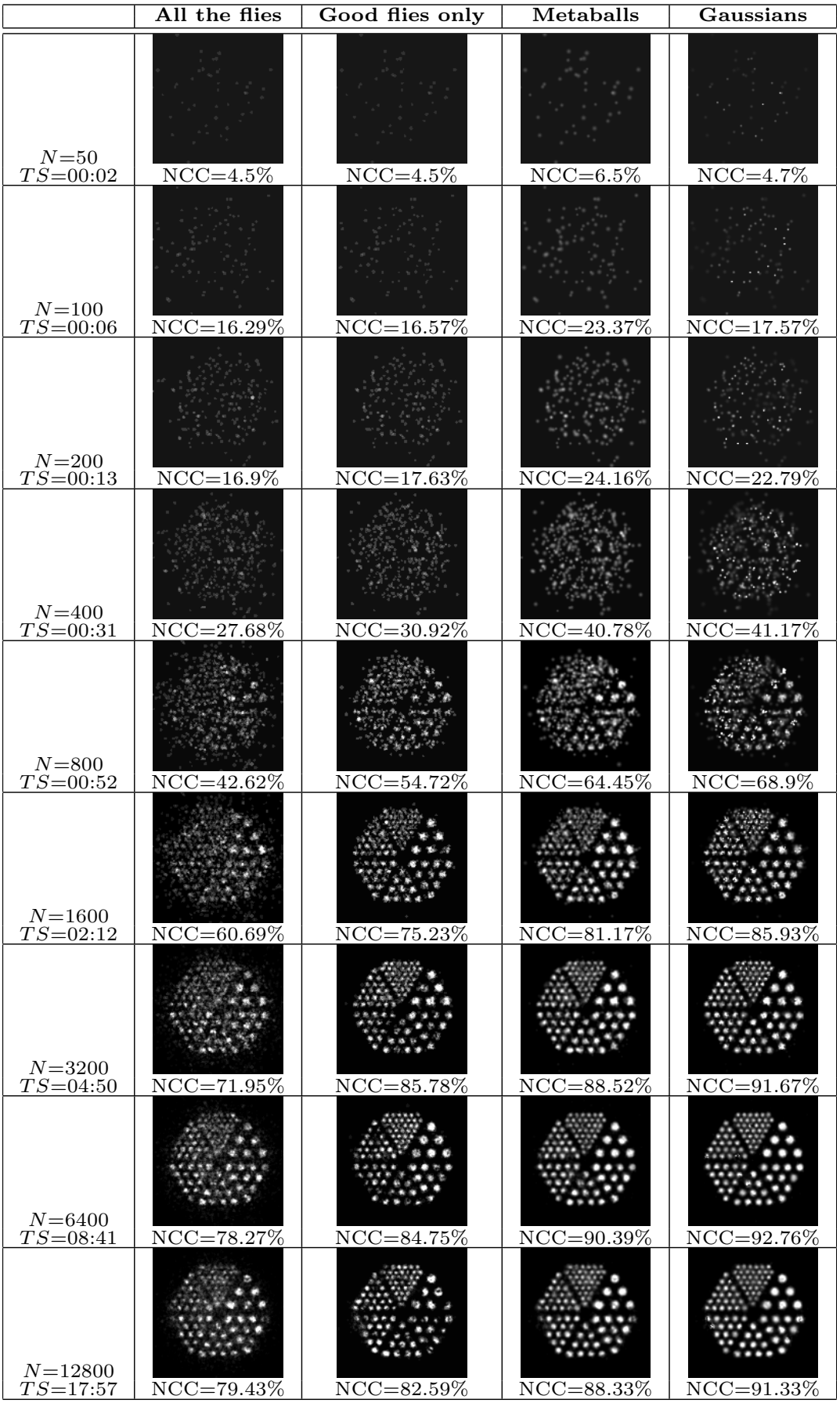
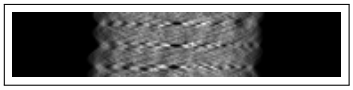


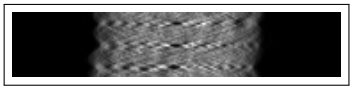
Figure 5.18: Evolutionary reconstructions of Figure 5.4a at successive resolutions (with N the number of flies and TS the time-stamp in minutes) using an Intel Xeon Westmere X5650 @ 2.67 GHz processor.

Using our naive approach as in [155, 154], the NCC would have been limited to 78.27% or 84.75% only. By exploiting the local fitness of flies to adapt Gaussian kernels, the image quality improves by about 10%. At similar levels of quality, the computing time is significantly reduced. For example, with our initial voxelisation method 17:57 minutes were required to obtain 82.59%. Only 2:12 minutes are needed to reach an even better NCC with adaptive Gaussian kernels.

5.5.2 Hot rode phantoms (low number of projections & noise)

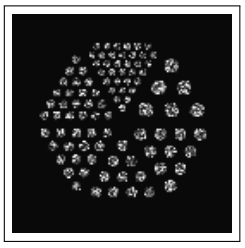


(a) Observations (known data).

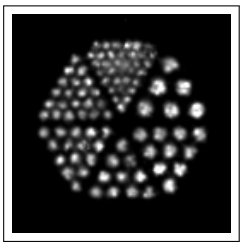


(b) Estimation (data simulated with all the flies) (NCC with (a): 99.63%).

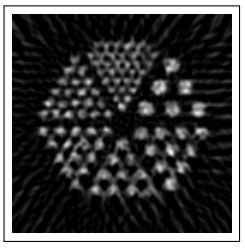
Figure 5.19: Sinograms from Figure 5.20a corresponding to the hot rode phantom with a low resolution and with noise. Images with 185 pixels per projection, 1st angle: 0°, angular step: 5°, and last angle: 175°.



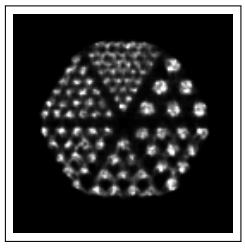
(a) Actual noisy phantom (NCC with ground-truth: 73.97%).



(b) Reconstruction with Gaussians (NCC with ground-truth: 85.92%).



(c) Reconstruction with FBP (NCC with ground-truth: 72.39%).



(d) Reconstruction with OSEM using 18 iterations and 2 subsets (NCC with ground-truth: 78.50%).

Figure 5.20: Tomographic reconstructions of the sinogram in Figure 5.19a corresponding to the hot rod example with a low number of angles and noise (see Figure 5.5a for the corresponding ground-truth).

To assess the algorithm in more difficult conditions, the number of angles in the sinogram is lowered and noise is included (see Figures 5.19 and 5.20a). The initial sinogram in Figure 5.4a was made of 180 rows with an angular step of 1°. The new sinogram contains 37 rows and the angular step is 5°. The sinogram estimated by the final population of flies (Figure 5.19b) is almost perfect (NCC of 99.63%). Reconstructions with FBP and OSEM are also considered as this test case is more realistic due to the noise in the input sinogram. Figure 5.20 shows the corresponding reconstructions. Our

Table 5.2: NCC between the ground-truth (Figure 5.5a) and the reconstructions of Figure 5.20. Numerical values in bold characters are the ones closest to the ground-truth.

Reconstruction type	NCC
All flies	79.79% ± 1.13%
Good flies only	78.94% ± 1.26%
Metaballs	83.82% ± 1.06%
Gaussian	**85.92% ± 0.60%**
FBP	72.39%
OSEM	78.50%

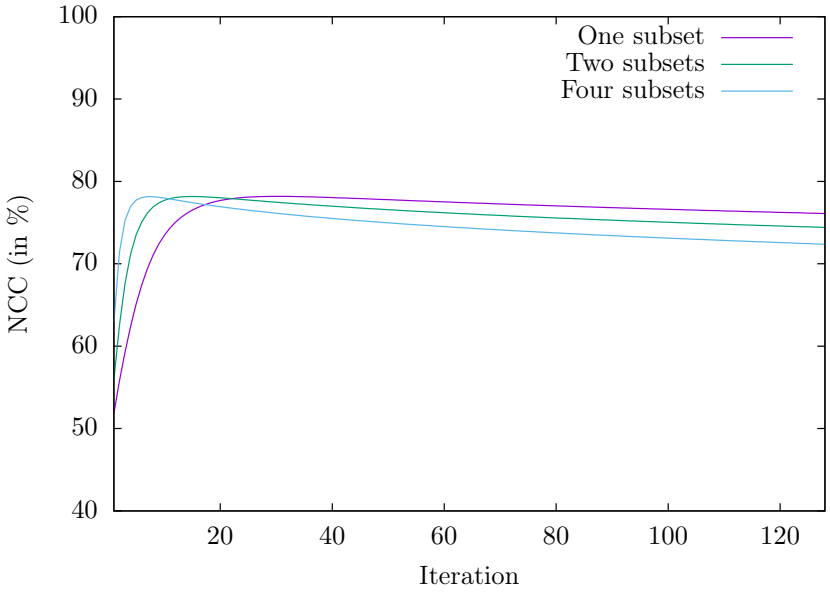


Figure 5.21: Hot rode phantoms (low number of projections & noise): Evolution of the NCC values between the ground-truth and images reconstructed at successive iterations of the OSEM method.

evolutionary reconstruction combined with our voxelisation method based on implicit modelling seem to provide more visually accurate results in this case than FBP and OSEM.

For a quantitative evaluation, Table 5.2 summarises the NCC values between the ground-truth (see Figure 5.5a) and the reconstructions of Figure 5.20. It shows that the Fly algorithm with density fields, both with Metaballs and Gaussian kernels, outperforms the traditional FBP and OSEM algorithms when the input data is of low resolution and noisy. One of the reasons of the outstanding performance of the Fly algorithm in the presence of noise is the stochastic nature of artificial evolution.

Figure 5.21 shows the NCC values between the ground-truth and images reconstructed using the OSEM algorithms with 1, 2, and 4 subsets for 128 successive iterations.

The NCC quickly increases then slowly decreases. Automatically determining how to stop the OSEM algorithm is still an open research question. Figure 5.22 presents our evolutionary reconstructions at successive resolutions for this test case. It shows the same pattern as previously observed. The algorithm is relatively quick to converge, then it slows done and stagnate. Only 40 seconds are required to reach a NCC of 80% with implicit modelling. In 7 minutes this threshold is not reached with binning.

5.5.3 Cardiac phantoms (with noise)

In Figure 5.23a, we use another numerical phantom that is more anatomically realistic. It corresponds to cardiac PET data. Noise is included in the phantom (see Figure 5.23b) to produce the sinogram of Figure 5.24a. Again, the sinogram estimated by the final population of flies (Figure 5.24b) is almost perfect (NCC of 99.87%). FBP, OSEM and an evolutionary reconstruction are given in Figure 5.23. In this test case, the NCCs of all the reconstructions are within less that 2% from each other (see Table 5.3).

We also assess reconstructions generated using the OSEM algorithms with 1, 2, and 4 subsets for 128 successive iterations (see Figure 5.25). The NCC between the ground-truth and images reconstructed with OSEM quickly increases then slowly decreases. Figure 5.26 presents our evolutionary reconstructions at successive resolutions for this test case. It shows again the same pattern as previously observed. The algorithm is relatively quick to converge, then it slows done and stagnate. 3 minutes are needed to reach a NCC of 90% with implicit modelling; 28 minutes with binning.

5.6 Conclusion

In the research presented here, we addressed the complex problem of medical tomographic reconstruction using evolutionary computing, by transposing the Fly Algorithm technique originally developed in a stereo-vision context for robotics. In classical EAs at the end of the algorithm the best individual is extracted and considered the solution of the optimisation problem, while all the rest of the population is discarded.

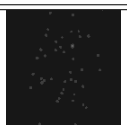
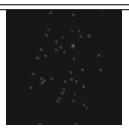
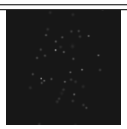
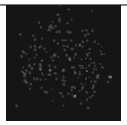
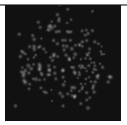
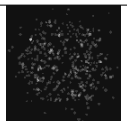
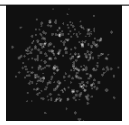
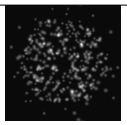
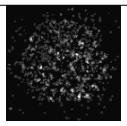
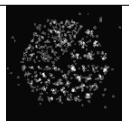
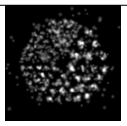
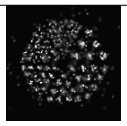
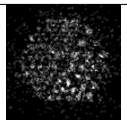
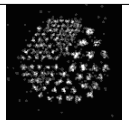
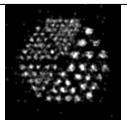
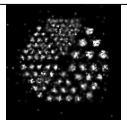
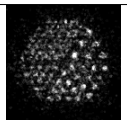
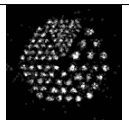
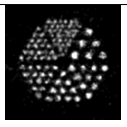
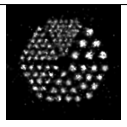
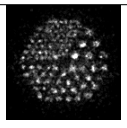
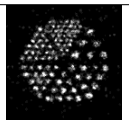
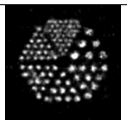
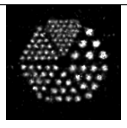
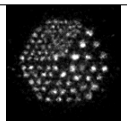
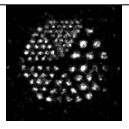
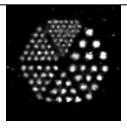
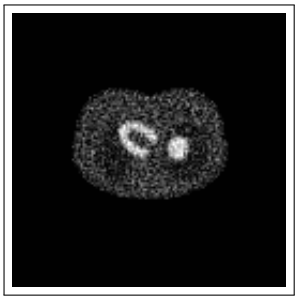
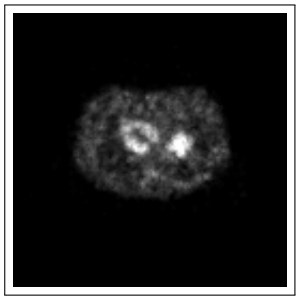
Number of flies	All the flies	Good flies only	Metaballs	Gaussians	Time-stamp (in MM:SS)
50	 NCC=8.82%	 NCC=8.82%	 NCC=11.83%	 NCC=7.39%	00:01
100	 NCC=9.97%	 NCC=9.97%	 NCC=15.10%	 NCC=13.93%	00:02
200	 NCC=17.34%	 NCC=18.37%	 NCC=25.24%	 NCC=22.98%	00:06
400	 NCC=29.92%	 NCC=35.13%	 NCC=4.14%	 NCC=37.49%	00:12
800	 NCC=45.71%	 NCC=58.52%	 NCC=67.77%	 NCC=64.56%	00:24
1600	 NCC=60.05%	 NCC=74.57%	 NCC=80.55%	 NCC=81.13%	00:42
3200	 NCC=68.65%	 NCC=80.16%	 NCC=84.27%	 NCC=86.00%	01:24
6400	 NCC=71.90%	 NCC=79.88%	 NCC=84.13%	 NCC=86.73%	03:21
12800	 NCC=74.40%	 NCC=77.73%	 NCC=83.42%	 NCC=86.94%	06:51

Figure 5.22: Evolutionary reconstructions of Figure 5.19a at successive resolutions.

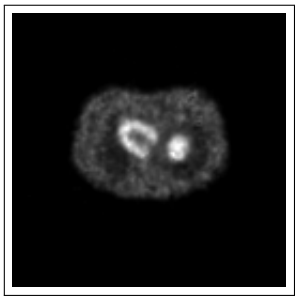
(a) Ground-truth.



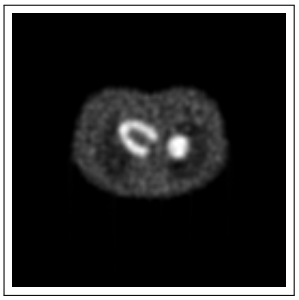
(b) Actual noisy phantom (NCC with (a): 91.12%).



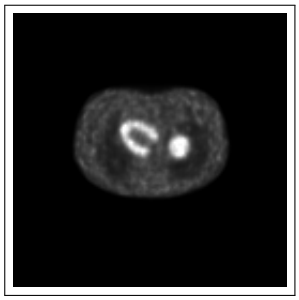
(c) Binning with all the flies (NCC with (a): 91.44%).



(d) Reconstruction with Gaussians (NCC with (a): 96.14%).



(e) Reconstruction with FBP (NCC with (a): 96.81%).



(f) Reconstruction with OSEM using 7 iterations and 4 subsets (NCC with (a): 97.07%).

Figure 5.23: Tomographic reconstructions of the sinogram in Figure 5.24a corresponding to the cardiac example, i.e. a more anatomically realistic sinogram with noise.

The Fly Algorithm, which is a cooperative co-evolution algorithm relies on a different philosophy, where each individual is a part of the solution. An individual corresponds to a 3-D point. The whole population is a representation of the reconstructed tomographic images. The evaluation of the performance of each individual is performed using a fitness function based on the leave-one-out-cross-validation method. If the fitness is positive, then the fly is improving the performance of the population; if it is negative, it is deteriorating the performance of the population. In the nuclear medicine context, the concentration of flies will be an estimation of the radioactive concentration.

In our previous developments we were only keeping the flies with a positive fitness. Binning (also call 'bucketing') was used to convert this point cloud into a discrete 3-D volume made of voxels: The space was divided in a regular 3-D grid and the voxel intensity corresponded to the number of good flies located into it. The local fitness of flies was not exploited during the voxelisation. No or very little comparison with traditional tomographic reconstruction algorithms in nuclear medicine was provided.

Table 5.3: NCC between the ground-truth and the reconstructions in the case of the cardiac example (Figure 5.23). Numerical values in bold characters are the ones closest to the ground-truth.

Reconstruction type	NCC
All flies	90.75% ± 0.92%
Good flies only	86.78% ± 1.63%
Metaballs	93.97% ± 0.68%
Gaussian	95.39% ± 0.45%
FBP	96.81%
OSEM	**97.07%**

(a) Observations (known data).

(b) Estimation (data simulated with all the flies) (NCC with (a): 99.87%).

Figure 5.24: Sinograms of the cardiac example from Figure 5.23a. Images with 185 pixels per projection, 1st angle: 0°, angular step: 1°, and last angle: 179°

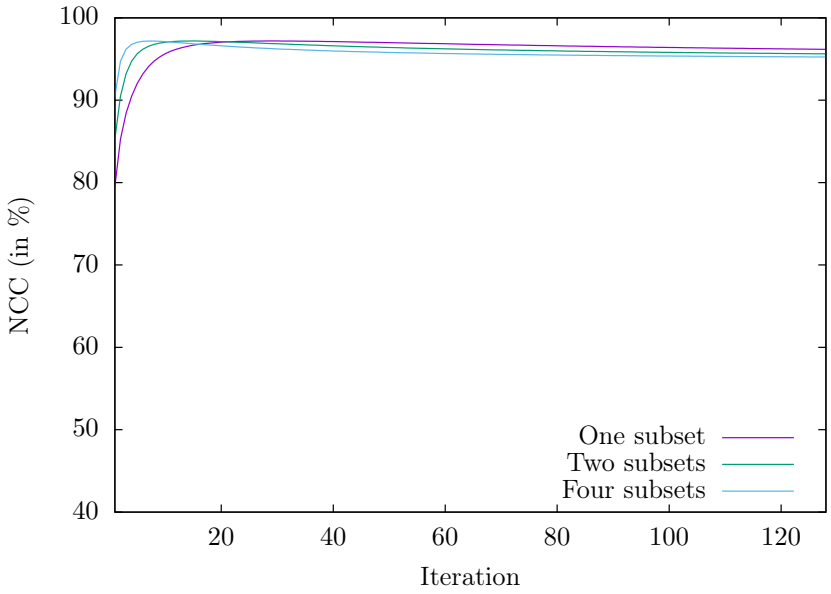


Figure 5.25: Cardiac example: Evolution of the NCC values between the ground-truth and images reconstructed at successive iterations of the OSEM method.

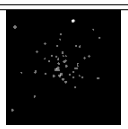
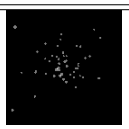
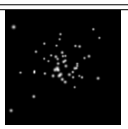
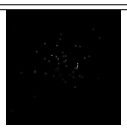
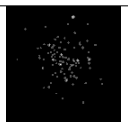
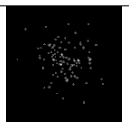
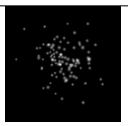
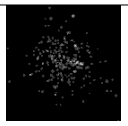
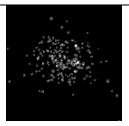
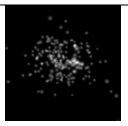
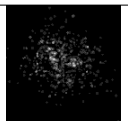
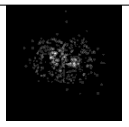
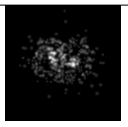
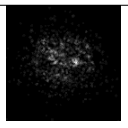
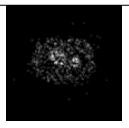
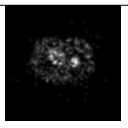
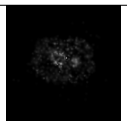
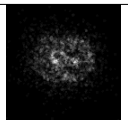
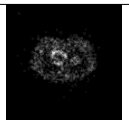
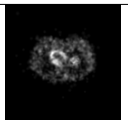
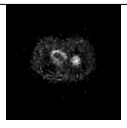
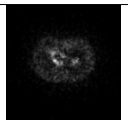
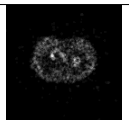
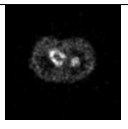
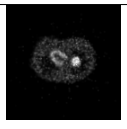
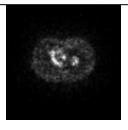
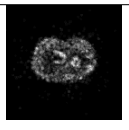
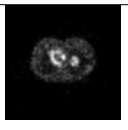
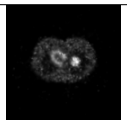
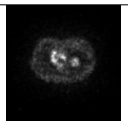
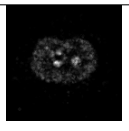
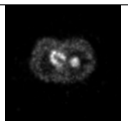
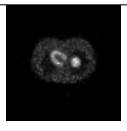
Number of flies	All the flies	Good flies only	Metaballs	Gaussians	Time-stamp (in MM:SS)
50	 NCC=21.18%	 NCC=22.94%	 NCC=34.03%	 NCC=23.64%	00:03
100	 NCC=36.01%	 NCC=37.40%	 NCC=52.24%	 NCC=43.52%	00:10
200	 NCC=44.11%	 NCC=48.59%	 NCC=64.38%	 NCC=58.84%	00:21
400	 NCC=56.42%	 NCC=63.89%	 NCC=76.35%	 NCC=73.65%	00:46
800	 NCC=72.45%	 NCC=77.55%	 NCC=87.58%	 NCC=86.88%	01:29
1600	 NCC=78.80%	 NCC=82.07%	 NCC=90.47%	 NCC=91.03%	03:10
3200	 NCC=86.80%	 NCC=78.38%	 NCC=92.54%	 NCC=93.47%	07:25
6400	 NCC=89.36%	 NCC=78.20%	 NCC=92.99%	 NCC=94.57%	16:30
12800	 NCC=90.14%	 NCC=81.31%	 NCC=93.47%	 NCC=95.94%	28:06

Figure 5.26: Evolutionary reconstructions of Figure 5.24a at successive resolutions.

We saw in this chapter that keeping the good flies only does not necessarily lead to the best quantitative results: Retaining the flies with a negative fitness yields more accurate results. The natural output of the Fly algorithm is a population of 3-D points. Here, we also exploit the point cloud one step further and use implicit modelling to voxelise the data from a density field. To this end, this research has investigated the use of Metaballs and Gaussian kernels - where the height and width of each Gaussian is computed to take into account the corresponding fly's performance. During the evolution, flies are discrete (from a numerical point of view) particle emitters: Flies are trying to replicate the observed data (what actually happened). At the end of the reconstruction, i.e. after convergence and during the extraction of the solution, each fly is considered as a given realisation of a stochastic process: A fly is an approximation of a random variable and, as such, can be modelled as a density field. It is actually intuitive to prefer implicit modelling over data binning as this is these stochastic events that we are actually trying to estimate. The marginal fitness can be considered as a confidence level in the fly's position. We demonstrate here that it is possible to take advantage of the fitness of each individual after the optimisation process to modulate the spread of flies depending on their respective performance. It improves quantitative results in all our test-cases by more than 10% in term of NCC compared to binning. A more accurate reconstruction is also achieved using less computational power. In this chapter, our reconstructions are also compared with those of FBP and OSEM, which are traditionally used in nuclear medicine. Using implicit modelling is used for voxelisation, more accurate results can be achieved in a much smaller amount of time than using binning in all our test-cases. Results show that density fields using Gaussian kernels lead to similar or better reconstructions than OSEM in term of numerical accuracy depending on the input data.

One key benefit of our method is its reliability: When an optimum solution has been found, increasing the computing time does not alter the reconstruction accuracy in a negative way as in OSEM. However, more work is needed to fully evaluate the method to better understand its limitations and how they compare with OSEM. More clinically realistic data would be required. The behaviour of the algorithm with respect to different amount of noise levels needs also to be evaluated. Future work will also include investigating the use of the latest advances in signal processing such as compressed sensing (also known as compressive sensing, compressive sampling, or sparse sampling).

The aim would be to increase the accuracy (reducing noise in the reconstructed images, whilst preserving edges), and also to reduce the computing time. The use of implicit modelling has also the potential to increase the pixel resolution of reconstructed images. Indeed, the sampling rate can be altered in Eq. 5.10 to increase the number of voxels in the output without introducing noise. Although it would need to be validated and compared with similar reconstructions by OSEM.

Chapter 6

Mutation Operators

6.1 Introduction

This chapter builds on the previous chapter. Our work is based on a Cooperative Co-evolution Algorithm – the Fly algorithm – in which individuals correspond to 3-D points. The Fly algorithm relies on mutation operators and a new blood operator to ensure diversity in the population. Our method heavily relies on the selection process, mutation operators and a diversity mechanism to find the best individuals' position. A large mutation variance is often initially used to avoid local maxima, and then progressively reduced to refine the results. Another approach is the use of adaptive operators. However, very little research on adaptive operators in the Fly algorithm has been conducted. We address this deficiency and propose 4 different fully adaptive mutation operators in the Fly algorithm: Basic Mutation, Adaptive Mutation Variance, Dual Mutation, and Directed Mutation. Due to the complex nature of the search space (kN-dimensions, with k the number of genes per individuals and N the number of individuals in the population), we favour operators with a low maintenance cost in term of computations. Their impact on the algorithm efficiency is analysed and validated on PET reconstruction.

We favour self-adaptive schemes keeping in mind that one of the main requirements is to minimise the administration cost in term of calculation. In this chapter, we study the behaviour of various mutation operators in our CCEA based on the Fly algorithm in PET reconstruction. The general principles of the evolutionary reconstruction for PET reconstruction is given in Section 2.4. The study includes a new operator called *Directed Mutation*. Section 6.2 defines the mutation operators, which are used in

our implementation. The results of these operators are analysed in Section 6.3 using two controlled test cases: with/without noise. The chapter ends with a conclusion in Section 6.4.

6.2 Varying mutation operators in the Fly algorithm

Our implementation relies on mutation to create better flies. The aim of the mutation operators is to create new flies in the neighbourhood of good flies. Note that new blood is also used to preserve a minimum level of diversity in the population. The following steps are necessary to use a mutation operator:

1. A bad fly is selected using the Threshold Selection.
2. Its projections are removed from $\left(P\left[\hat{f}\right]\right)$.
3. A good fly is selected using the Threshold Selection.
4. The bad fly is replaced by the good fly.
5. The position of the newly created fly is altered by random changes.
6. The projections of the mutated fly are computed.
7. These projections are added to $\left(P\left[\hat{f}\right]\right)$.

The only step which is different, depending on the mutation operator used, is 5. Ideally, the amount of random change needs first to be set to a large value to better explore the search space. However, a constant large mutation variance will lead to blurred reconstructed volumes. As a consequence, the mutation variance has to be gradually reduced. The usefulness of adaptive mutations in evolutionary algorithms is a well established [12, 34, 50, 119]. Such techniques have been proven effective in various cases, depending on the fitness function and the genetic engine used. However, complex schemes for the adaptivity of the mutation operator have a computational cost that may not be negligible. More simplistic schemes can actually perform better due to

lower computational needs [41]. Our main motivation is to investigate the use of such operators in the Fly algorithm. The aim is to determine which sets of operators are the best in term of accuracy of the results, and amongst them which one is the best in term of computational cost. Destroying a bad fly and creating new and better ones has to be a fast process because it is performed at a much higher rate in the Parisian approach than in classical EAs. This is because the solution to the optimisation problem in our case is the whole population [7] rather than the best individual as in classical EAs. Using the best set of mutation operators to create new flies is therefore important.

In Vidal *et al* initial implementation, only the Dual mutation was used [155]. We added three other adaptive mutation operators that are automatically tuned without any human intervention. An iteration of the evolution loop is given in Algorithm 3. An individual has 9 genes:

1. x, the fly's position along the x-axis,
2. y, the fly's position along the y-axis,
3. z, the fly's position along the z-axis,
4. $P_{basicMut}$, the probability of the basic mutation operator,
5. $P_{adaptiveMut}$, the probability of the adaptive mutation operator,
6. $P_{dualMut}$, the probability of the dual mutation operator,
7. $P_{directedMut}$, the probability of the directed mutation operator,
8. P_{new_blood}, the probability of the immigration/new blood operator,
9. σ, the mutation rate associated with the fly (it is used by the basic and directed mutation operators).

The mutation operators are described below. They share the same procedure to apply the actual random changes (see Algorithm 4).

6.2.1 Basic mutation

The mutation variance can be subject to an adaptive pressure itself and be self-adapted [12]. In our implementation the probability of all the operators ($P_{basicMut}$, $P_{adaptiveMut}$, $P_{dualMut}$, $P_{directedMut}$, and P_{new_blood}) is encoded in the genome of each individual (see Algorithm 4 for the Procedure *mutate*). The mutation variance (σ) is too. The probabilities and the variance are then subject to random mutations as well. The major advantage of this scheme is to provide a fully automatic method to adapt the mutation variance, whilst keeping the administration cost null.

Algorithm 3 Simplified evolutionary loop focusing on the mutation operators

```
# Initialisation of internal states
N ← population size
σ_min ← 1%,   σ_max ← 20%
k ← 2^(1/3)
period ← N
σ_high ← 10%, σ_low ← σ_high/k
counter_σ_high ← counter_σ_low ← Δσ_high ← Δσ_low ← 0

# Evolutionary loop
repeat
  # Get the min/max local fitness values
  fit_min ← min local fitness of population
  fit_max ← max local fitness of population

  # Create N new individuals
  for i = 0 to N do

    old_global_fitness ← Compute the global fitness

    # Find a bad fly (fly⁻) and a good fly (fly⁺)
    fly⁻ ← Select randomly a fly from the population. fly⁻'s local fitness must
be negative
    fly⁺ ← Select randomly a fly from the population. fly⁺'s local fitness must
be positive

    Kill fly⁻ # Remove its contribution from simulated_image

    genetic_operator ← Choose a random number between 0 and 1
```

Use genetic operator probabilities provided by fly^+

# Dual mutation
if $genetic\_operator \leq fly^+.P_{dualMut}$ **then**
$new\_fly \leftarrow$ mutation(fly^+, getDualMutationRate(σ_{low}, σ_{high}, $counter\_\sigma_{low}$, $counter\_\sigma_{high}$), FALSE, σ_{min}, σ_{max})

Basic mutation
else if $genetic\_operator \leq (fly^+.P_{dualMut} + fly^+.P_{basicMut})$ **then**
$new\_fly \leftarrow$ mutation(fly^+, $fly^+.\sigma$, FALSE, σ_{min}, σ_{max})

Adaptive mutation
else if $genetic\_operator \leq (fly^+.P_{dualMut} + fly^+.P_{basicMut} + fly^+.P_{adaptiveMut})$ **then**
$new\_fly \leftarrow$ mutation(fly^+, getAdpativeMutationRate(fly^+.localFitness(), fit_{min}, fit_{max}, σ_{min}, σ_{max}), FALSE, σ_{min}, σ_{max})

Directed mutation
else if $genetic\_operator \leq (fly^+.P_{dualMut} + fly^+.P_{basicMut} + fly^+.P_{adaptiveMut} + fly^+.P_{directedMut})$ **then**
$new\_fly \leftarrow$ mutation(fly^+, $fly^+.\sigma$, TRUE, σ_{min}, σ_{max})

New blood / Immigration
else if $genetic\_operator \leq (fly^+.P_{dualMut} + fly^+.P_{basicMut} + fly^+.P_{adaptiveMut} + fly^+.P_{directedMut} + fly^+.P_{new\_blood})$ **then**
$new\_fly \leftarrow randomIndividual()$
end if

checkGeneRange(new_fly) # Make sure the gene values of the new fly are valid

Replace fly^- by new_fly # Add new_fly's contribution to $simulated\_image$

$new\_global\_fitness \leftarrow$ Compute the global fitness

# Update some states for dual mutation if needed
if $genetic\_operator \leq fly^+.P_{dualMut}$ **then**

Record the performance of σ_{high} or σ_{low} on the global fitness
Call updateDualMutationData($old\_global\_fitness$, $new\_global\_fitness$, $counter\_\sigma_{low}$, $counter\_\sigma_{high}$, $\Delta\sigma_{low}$, $\Delta\sigma_{high}$)

Update σ_{high} and σ_{low} if needed
Call updateDualMutationRate(k, $period$, σ_{low}, σ_{high}, $counter\_\sigma_{low}$, $counter\_\sigma_{high}$, $\Delta\sigma_{low}$, $\Delta\sigma_{high}$)
end if
end for
until Stopping criteria met
$simulated\_image$: the image simulated by the whole population.

Algorithm 4 Procedure mutate

Input: fly^+ # The good fly on which new_fly will be based
Input: σ # The σ value to use for small random alterations
Input: use_dir_mut # A boolean flag
Input: σ_{min} # The min possible value of σ
Input: σ_{max} # The max possible value of σ
Output: new_fly # The fly create by mutation of fly^+

$new\_fly.parentF_m = fly^+.F_m$ # Record the parent's fitness

if use_dir_mut is FALSE **then** # NOT Directed-Mutation
  # Use random directions
  $new\_fly.dir\,[0] \leftarrow \text{sign}(\text{rand}(-1.0, 1.0))$
  $new\_fly.dir\,[1] \leftarrow \text{sign}(\text{rand}(-1.0, 1.0))$
  $new\_fly.dir\,[2] \leftarrow \text{sign}(\text{rand}(-1.0, 1.0))$
  $new\_fly.dir\,[3] \leftarrow \text{sign}(\text{rand}(-1.0, 1.0))$
  $new\_fly.dir\,[4] \leftarrow \text{sign}(\text{rand}(-1.0, 1.0))$
  $new\_fly.dir\,[5] \leftarrow \text{sign}(\text{rand}(-1.0, 1.0))$
  $new\_fly.dir\,[6] \leftarrow \text{sign}(\text{rand}(-1.0, 1.0))$
  $new\_fly.dir\,[7] \leftarrow \text{sign}(\text{rand}(-1.0, 1.0))$
  $new\_fly.dir\,[8] \leftarrow \text{sign}(\text{rand}(-1.0, 1.0))$
else # Use Directed-Mutation
  if $fly^+.F_m > fly^+.parentF_m$ **then**
    # Parent better than grand-parent
    $\alpha \leftarrow 1$ # use the same directions as the parent
  else # Grand-parent better than parent
    $\alpha \leftarrow -1$ # Use the opposite directions as the parent
  end if
  # Use (opposite) directions of fly^+
  $new\_fly.dir\,[0] \leftarrow \alpha \times fly^+.dir\,[0]$
  $new\_fly.dir\,[1] \leftarrow \alpha \times fly^+.dir\,[1]$
  $new\_fly.dir\,[2] \leftarrow \alpha \times fly^+.dir\,[2]$
  $new\_fly.dir\,[3] \leftarrow \alpha \times fly^+.dir\,[3]$
  $new\_fly.dir\,[4] \leftarrow \alpha \times fly^+.dir\,[4]$
  $new\_fly.dir\,[5] \leftarrow \alpha \times fly^+.dir\,[5]$
  $new\_fly.dir\,[6] \leftarrow \alpha \times fly^+.dir\,[6]$
  $new\_fly.dir\,[7] \leftarrow \alpha \times fly^+.dir\,[7]$
  $new\_fly.dir\,[8] \leftarrow \alpha \times fly^+.dir\,[8]$
end if
# Add small random changes in the chosen directions
$new\_fly.x \leftarrow fly^+.x + new\_fly.dir\,[0] \times \text{rand}(0.0, 0.5) \times \sigma \times range_x$
$new\_fly.y \leftarrow fly^+.y + new\_fly.dir\,[1] \times \text{rand}(0.0, 0.5) \times \sigma \times range_y$
$new\_fly.z \leftarrow fly^+.z + new\_fly.dir\,[2] \times \text{rand}(0.0, 0.5) \times \sigma \times range_z$
$new\_fly.P_{dualMut} \leftarrow fly^+.P_{dualMut} + new\_fly.dir\,[3] \times \text{rand}(0.0, 0.5) \times \sigma$
$new\_fly.P_{basicMut} \leftarrow fly^+.P_{basicMut} + new\_fly.dir\,[4] \times \text{rand}(0.0, 0.5) \times \sigma$
$new\_fly.P_{adaptiveMut} \leftarrow fly^+.P_{adaptiveMut} + new\_fly.dir\,[5] \times \text{rand}(0.0, 0.5) \times \sigma$
$new\_fly.P_{directedMut} \leftarrow fly^+.P_{directedMut} + new\_fly.dir\,[6] \times \text{rand}(0.0, 0.5) \times \sigma$
$new\_fly.P_{new\_blood} \leftarrow fly^+.P_{new\_blood} + new\_fly.dir\,[7] \times \text{rand}(0.0, 0.5) \times \sigma$
$new\_fly.\sigma \leftarrow fly^+.\sigma + new\_fly.dir\,[8] \times \text{rand}(0.0, 0.5) \times \sigma \times (\sigma_{max} - \sigma_{min})$

6.2.2 Adaptive mutation variance

The mutation variance can be directly adapted to local measurements, like fitness [120] or local regularity [108]. Another approach, called Rechenberg's rule, is to modulate the mutation variance based on the success/failure rate of the current mutation variance [133, 18]. It relies on the notion of "evolution window": Increase the mutation variance to speed-up the search-space exploration, or decrease it to refine the results. For this purpose, the algorithm must keep track of the success rate, which has an obvious computational cost. To reduce the administration cost of the algorithm, in [158] the variance is bigger when fitness is high and smaller when fitness is low. The evolutionary algorithm was used to minimise an error function. The idea was to favour large exploration around the weakest individuals, whilst performing fine tuning in the vicinity of good individuals. In our case, we want to maximise the marginal fitness of flies: The higher the marginal fitness (F_m), the lower the variance, and *vice versa*. We define the mutation variance here as a piecewise-defined function of F_m:

$$\sigma\left(F_m\right) = \begin{cases} \sigma_{max}, & F_m < fit_{min} \\ \sigma_{min}, & F_m > fit_{max} \\ \sigma_{min} + \left(\sigma_{max} - \sigma_{min}\right) \times \frac{\cos\left(\pi \times \left(\frac{F_m - fit_{min}}{fit_{max} - fit_{min}}\right)\right) + 1.0}{2.0}, & \text{otherwise} \end{cases}$$

F_m corresponds to the fitness of the individual who will undergo a mutation. $\sigma(F_m)$ smoothly varies between the smaller (fit_{min}) and the larger (fit_{max}) fitness thresholds respectively. If F_m is smaller than fit_{min}, σ is then σ_{max}; if the individual's fitness is greater than fit_{max}, σ is then σ_{min} (with σ_{min} and σ_{max} two constant values set by the user). The major advantage of this scheme is similar to the previous one. It provides a fully automatic method to adapt the mutation variance, whilst keeping the administration cost of the algorithm negligible (see Algorithm 5 for the corresponding procedure).

Algorithm 5 Procedure getAdpativeMutationRate

Input: F_m # The local fitness function (measure to maximise)
Input: fit_{min}
Input: fit_{max}
Input: σ_{min}
Input: σ_{max}
Output: σ

if $F_m < fit_{min}$ **then#** Use the higher value of σ for very bad flies
  $\sigma \leftarrow \sigma_{max}$
else if $F_m > fit_{max}$ **then#** Use the lower value of σ for very good flies
  $\sigma \leftarrow \sigma_{min}$
else # Use the compute value of σ for intermediate flies
  $\sigma \leftarrow \sigma_{min} + (\sigma_{max} - \sigma_{min}) \times \frac{\cos\left(\pi \times \left(\frac{F_m - fit_{min}}{fit_{max} - fit_{min}}\right)\right) + 1.0}{2.0}$
end if

6.2.3 Dual mutation

The variance can be tuned depending on some success measurement. This category includes the well-known 1/5$^{\text{th}}$ rule proposed by Schewefel [138, 18]. A single σ value is used. It is updated at regular intervals. It records the number of successful and unsuccessful mutations over a given number of mutations. If the rate of successful mutation is greater than $1/5$, then increase σ; if it is lower, decrease σ.

In [155], Vidal *et al.* proposed the *Dual mutation* operator. Our main goal was to provide a self-adaptive operator to limit the level of human input. It is based on the concurrent testing of two alternative variance values (σ_{low} and σ_{high}, with $k\sigma_{low} = \sigma_{high}$). Its pseudocode is available in Algorithms 6 and 7. The update rule is multiplicative as for the 1/5$^{\text{th}}$ rule. If mutations with σ_{high} provide the best results during the previous $period$ iterations, then both mutation variances are multiplied by a predefined factor (pf, with $pf > 1$) (see Algorithm 8). If mutations with σ_{low} provide the best results during the previous $period$ iterations, then both mutation variances are divided by pf. For every Dual mutation, we check the global fitness before and after the mutation. Note that these numbers are pre-computed in any case during the selection of individuals. Therefore, we can not affect their computation to the administration cost of this mutation operator. Using two accumulators, we can assess which variance amongst σ_{low} and σ_{high} is the best. This scheme is also providing a method with a very limited number of user inputs. The administration cost of the algorithm is sightly increased but still

Algorithm 6 Procedure getDualMutationRate

Input: σ_{low}
Input: σ_{high}
Input: $counter\_\sigma_{low}$
Input: $counter\_\sigma_{high}$
Output: σ

if $counter\_\sigma_{high} \leq counter\_\sigma_{low}$ **then**
  # Use the higher value of σ
  $\sigma \leftarrow \sigma_{high}$
else # Use the lower value of σ
  $\sigma \leftarrow \sigma_{low}$
end if

Algorithm 7 Procedure updateDualMutationData

Input: $old\_global\_fitness$ # Global fitness before dual-mutation
Input: $new\_global\_fitness$ # Global fitness after dual-mutation
Input/Output: $counter\_\sigma_{low}$
Input/Output: $counter\_\sigma_{high}$
Input/Output: $\Delta\sigma_{low}$
Input/Output: $\Delta\sigma_{high}$

$\Delta \leftarrow new\_global\_fitness - old\_global\_fitness$

if $counter\_\sigma_{high} \leq counter\_\sigma_{low}$ **then**
  # Use the higher value of σ
  $\Delta\sigma_{high} \leftarrow \Delta\sigma_{high} + \Delta$
  $counter\_\sigma_{high} \leftarrow counter\_\sigma_{high} + 1$
else # Use the lower value of σ
  $\Delta\sigma_{low} \leftarrow \Delta\sigma_{low} + \Delta$
  $counter\_\sigma_{low} \leftarrow counter\_\sigma_{low} + 1$
end if

relatively light. Also, the dual mutation does not need to make any assumption on the ideal success rate of the mutation as in the $1/5^{\text{th}}$ rule.

6.2.4 Directed mutation

We introduce here a new operator, the *Directed Mutation*. It is related to the evolution path in CMA-ES [72] Its objective is to lead new individuals toward areas of the search space that have been previously defined as "interesting" by older flies. This principle follows well the fundamentals of CCEAs as new individuals have to cooperate with older and wiser ones to benefit from their knowledge to locate areas of interest.

Algorithm 8 Procedure updateDualMutationRate

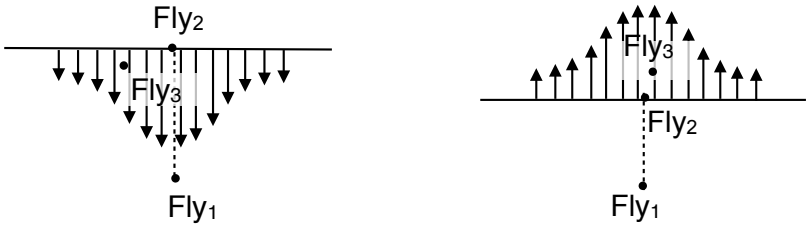
Input: k
Input: $period$
Input/Output: σ_{low}
Input/Output: σ_{high}
Input/Output: $counter\_\sigma_{low}$
Input/Output: $counter\_\sigma_{high}$
Input/Output: $\Delta\sigma_{low}$
Input/Output: $\Delta\sigma_{high}$

```
# Update needed
if period ≥ (counter_σ_high + counter_σ_low then

    # Compute the average score for each σ values
    score_σ_high ← Δσ_high/counter_σ_high
    score_σ_low ← Δσ_low/counter_σ_low

    if score_σ_high > score_σ_low then
        # The lowest parameter provides the best results
        σ_high ← σ_low
        σ_low ← σ_low/k
    else # The highest parameter provides the best results
        σ_low ← σ_high
        σ_high ← σ_high × k
    end if

    # Reset states to initial value
    counter_σ_high ← 0
    counter_σ_low ← 0
    Δσ_high ← 0
    Δσ_low ← 0
end if
```

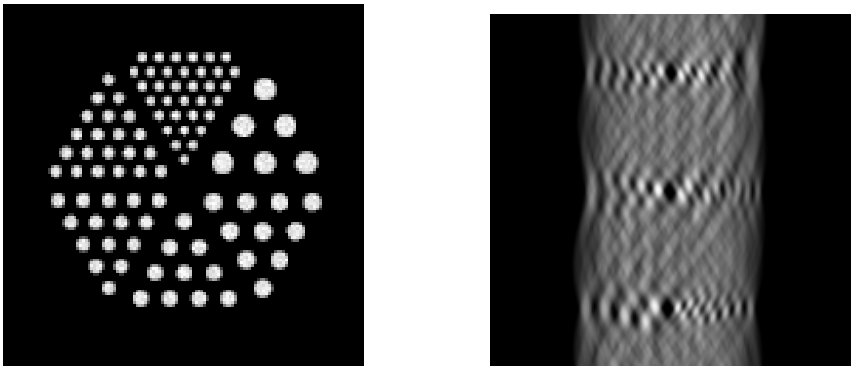


(a) Fly_1 better than Fly_2. **(b)** Fly_2 better than Fly_1.

Figure 6.1: Directed Mutation Principle.

To illustrate how our implementation works, let us consider the case as follows: A fly (Fly_2) has been created by mutation of another fly (Fly_1). We are now going to create a new fly (Fly_3) by mutation of Fly_2. The position of the new fly will be biased toward the position of the best fly among Fly_1 and Fly_2. If Fly_1 is better than Fly_2, we will look for a new Fly_3 from the location of Fly_2 in the direction toward Fly_1 (see Fig. 6.1a); if Fly_2 is better than Fly_1, we will look for Fly_3 from the location of Fly_2 in the direction away from Fly_1 (see Fig. 6.1b). For any fly created by any kind of mutation, we record its parent's fitness and in which direction the new fly has been moved with respect to its parent (see Algorithm 4). This is the main administration cost of our new operator. This is required due to the steady state nature of the algorithm. When Fly_3 is created, it is possible that Fly_1 has already been replaced by another fly.

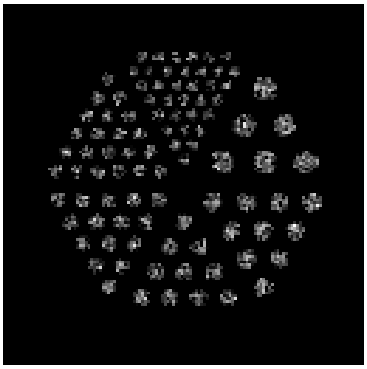
6.3 Results



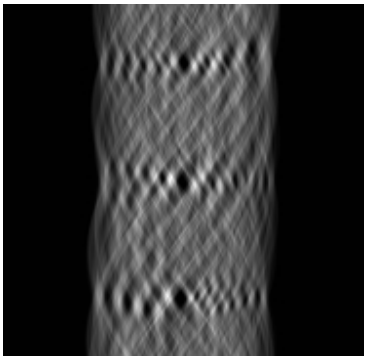
(a) Ground-truth. **(b)** Measured projections.

Figure 6.2: Test case using the Jaszczak phantom with hot rods.

For testing purposes, we consider the Jaszczak phantom with hot rods without and with noise (see Figures 6.2 and 6.3 respectively). We test the algorithm with all the possible combinations of mutation operators. There are 4 mutation operators,



(a) Simulated phantom.



(b) Measured projections.

Figure 6.3: Similar test case as Figure 6.2 but with noise.

therefore there are 2^4 possible configurations (see Tab. 6.1). Due to the stochastic nature

Table 6.1: The combinations of mutation operators.

Type	Operators	Type	Operators
0000	no mutation	1000	basic
0001	directed	1001	basic + directed
0010	adaptive	1010	basic + adaptive
0011	adaptive + directed	1011	basic + adaptive + directed
0100	dual	1100	basic + dual
0101	dual + directed	1101	basic + dual + directed
0110	dual + adaptive	1110	basic + dual + adaptive
0111	dual + adaptive + directed	1111	basic + dual + adaptive + directed

of artificial evolution, 15 reconstructions per configuration are performed to gather statistically meaningful results. It leads to 240 reconstructions per test case, which means that 480 evolutionary reconstructions in total have been performed. For each reconstruction, we record i) the NCC (see Equation 5.4) between the ground-truth (I_1) and the reconstructed volume (I_2), and ii) the reconstruction time. The NCC is 100% if the two images are perfectly correlated. It is 0% is they are totally uncorrelated. It is -100% if there is a negative correlation (also called anticorrelation or inverse correlation) between them.

6.3.1 Without noise in the input data

Figure 6.4 shows the median results in term of performance for duration and NCC for each mutation operator combination. We can see that the dual mutation combined with the directed mutation (see Configuration Type 0101 in Table 6.1) is apparently effective.

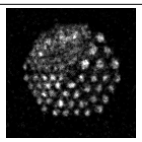
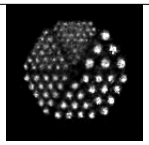
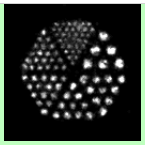
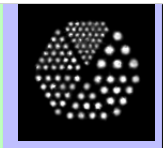
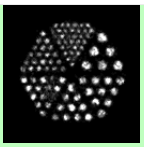
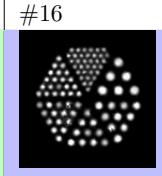
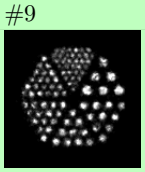
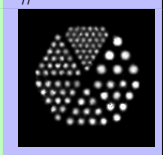
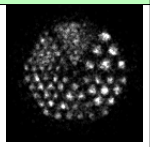
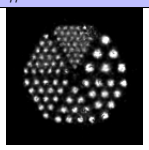
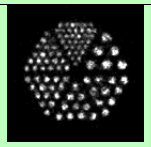
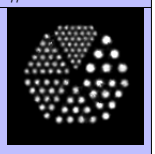
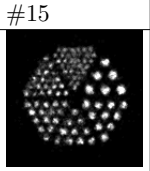
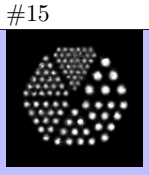
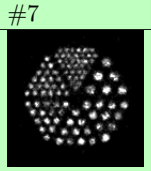
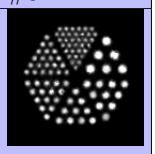
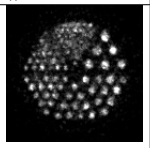
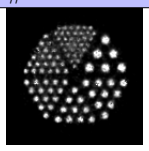
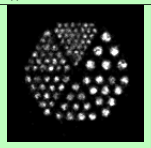
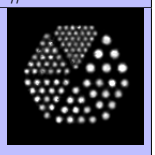
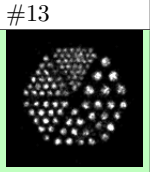
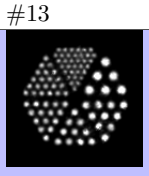
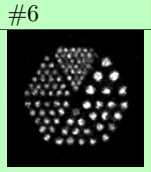
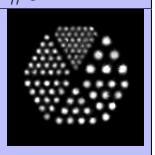
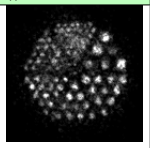
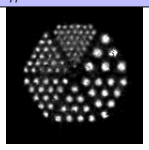
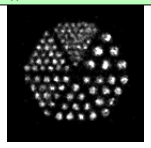
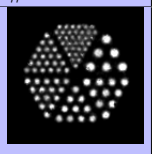
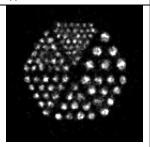
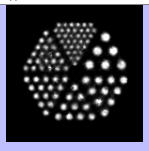
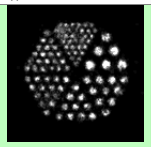
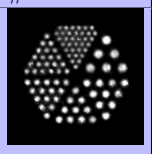
Type/mm #rank	Dirac #rank	Gaussian #rank	Type/mm #rank	Dirac #rank	Gaussian #rank
0000/14.53 #1	 (68.77%) #16	 (84.38%) #16	1000/15.40 #3	 (85.29%) #9	 (92.14%) #12
0001/15.67 #5	 (86.25%) #2	 (92.56%) #10	1001/15.67 #5	 (86.06%) #3	 (92.17%) #11
0010/14.53 #1	 (74.28%) #15	 (88.67%) #15	1010/16.00 #8	 (85.59%) #7	 (92.81%) #3
0011/18.93 #16	 (84.56%) #12	 (92.65%) #7	1011/17.80 #14	 (86.01%) #4	 (92.99%) #1
0100/15.67 #5	 (77.42%) #13	 (90.05%) #13	1100/16.73 #11	 (85.76%) #6	 (92.60%) #8
0101/16.40 #10	 (86.28%) #1	 (92.96%) #2	1101/16.73 #11	 (85.77%) #5	 (92.73%) #5
0110/15.40 #3	 (74.41%) #14	 (88.71%) #14	1110/16.13 #9	 (84.68%) #11	 (92.57%) #9
0111/16.87 #13	 (84.79%) #10	 (92.68%) #6	1111/17.87 #15	 (85.42%) #8	 (92.80%) #4

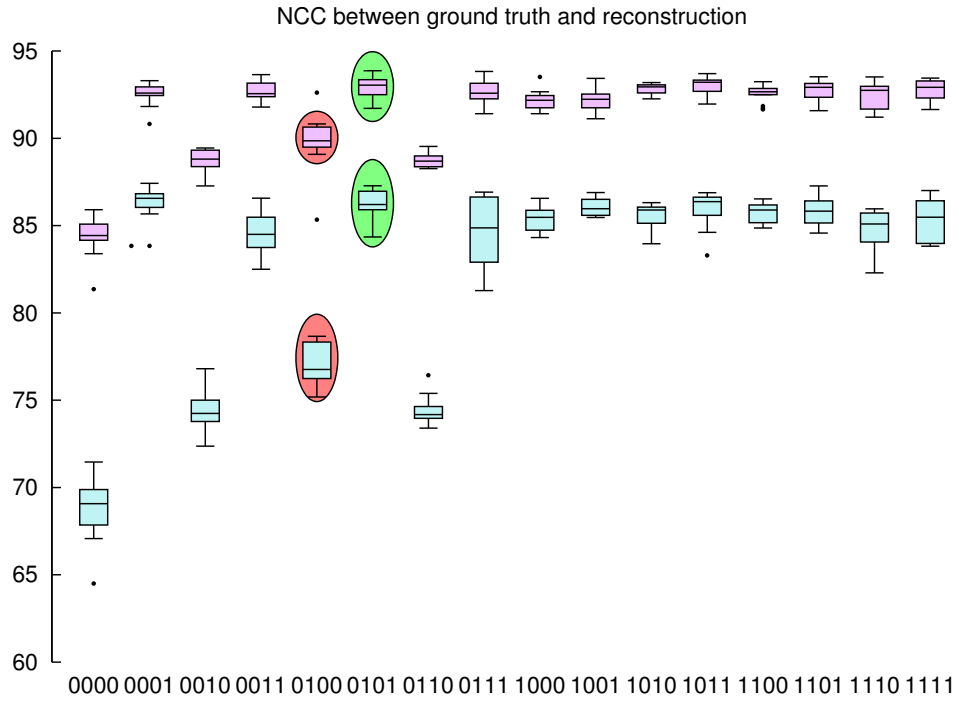
Figure 6.4: Performance comparison of the different combinations of mutation operators. Highlighted in green and blue-violet are the combinations whose NCC is less than 1% smaller than the best combination.

The dual mutation only (0100) as in [155] is not so good. In green and blue-violet are highlighted in the figure the combinations whose NCC is less than 1% smaller than the best combination. We can see that all of the combinations using the directed mutation are doing quite well. Only combinations that are shown in both green and blue-violet should be considered (i.e. Configuration 0001, 0101, 1000, 1001, 1010, 1011, 1100, 1101 and 1111). Ideally, the ones with a short run-time should be favoured.

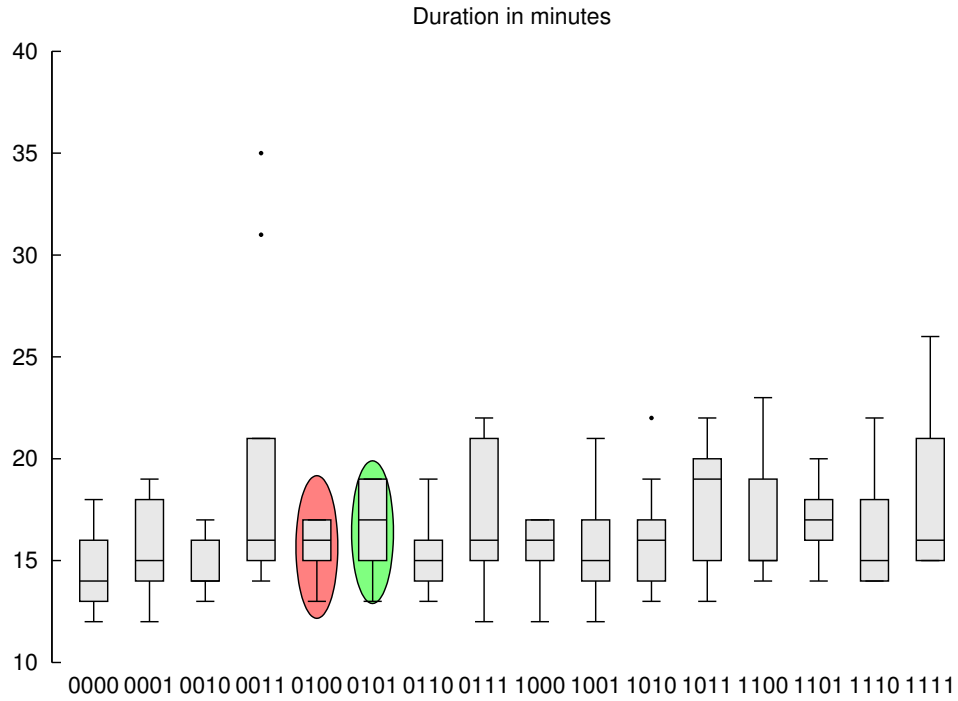
However, it is still not trivial to determine which configuration is the best one. A configuration can be considered good if it has a relatively large NCC, a small duration, and small standard deviations over the 15 runs for both NCC and duration. Quantitative results for each configuration are presented in Figure 6.5 using boxplots. The dark turquoise and purple configurations of Figure 6.5a shows the NCC between the ground-truth and the reconstructions using all flies as a finite point (or Dirac) and all flies using Gaussian kernel, respectively. The experiments with the dual and directed mutation operator (0101) (green circle) seem to provide best results (86.28 ±0.71) and (92.96 ±0.67) in terms of NCC in both configurations. It is much better than with the Dual mutation only (0100) (red circle) when we use all flies as finite points (77.42% ±2.41) as in [155] and all flies as Gaussian kernels (90.05 ±0.89). However it is still hard to assess which configuration is the best in term of reconstruction speed (see Figure 6.5b).

For a more reliable visual interpretation, we consider the average and standard deviation for NCC and duration for all the combinations of operators. The data is plotted in Figure 6.6a and 6.6b using ellipses. Each ellipse corresponds to a combination of operators. They are centred on their respective average of NCC and duration. Their radii correspond to the standard deviations. The best combination corresponds to an ellipse with small radii and that is located in the top left corner of the figure. This validates our initial assumption that the dual and directed mutation (0101) is one of the best combinations in term of NCC and acceptable in terms of computation time (16.4 minutes ± 2.29).

From our visual observations, Configuration 0101 seems to be good. It actually has the highest average NCC and its standard deviation is small. Our hypothesis is that 0101 is the best combination of operators. To validate this hypothesis, we apply a non-parametric statistical hypothesis test (Wilcoxon signed-rank test) and presents the

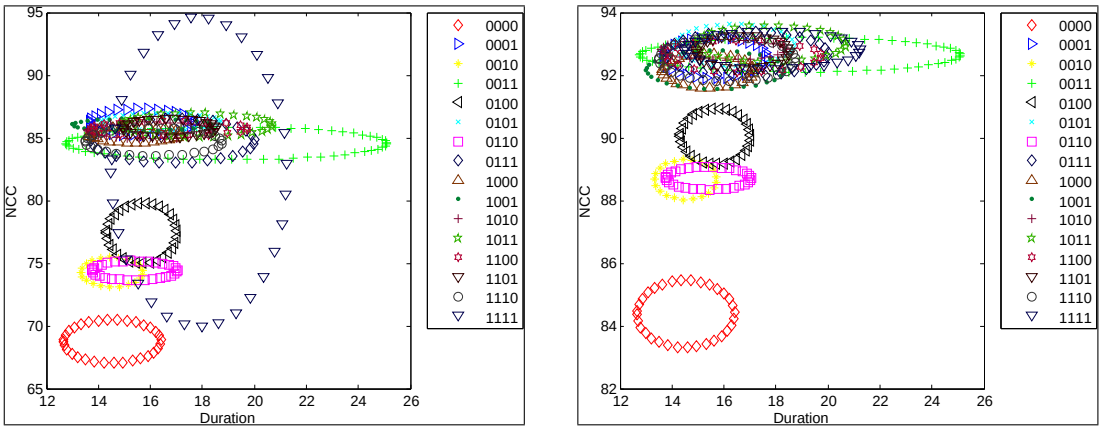


(a) NCC for all flies as a finite point (in blue) and all flies as Gaussian kernel (in magenta) [7].



(b) Duration.

Figure 6.5: Performance of mutation operators. In red is highlighted the performance of our initial implementation with dual mutation only as in [155]. In green is highlighted the performance of the combination of the dual and directed mutation operators.



(a) All the flies **(b)** All the flies with Gaussian kernel

Figure 6.6: Average and standard deviation of NCC and duration (without noise in the input data).

Table 6.2: Performance comparison in terms of NCC of Combination 0101 with all the other combinations of mutation operators with all flies as Gaussian kernel for the reconstructions with/without noise in the input data. Combinations with p-value higher than 0.05 for both without noise and with noise are highlighted in pink. Other p-values higher than 0.05 for one of the test case only are highlighted in green.

	Gaussian without noise		**Gaussian with noise**	
Combination type	**NCC (in %)**	Wilcoxon (p-value)	**NCC (in %)**	Wilcoxon (p-value)
0000	84.38 ±1.07	0.00064	81.53 ±0.84	0.00064
0001	92.556 ±0.65	0.11184	83.30 ±0.93	0.39532
0010	88.67 ±0.64	0.00064	80.76 ±0.56	0.00064
0011	92.65 ±0.55	0.2113	83.45 ±0.94	0.56868
0100	90.05 ±0.89	0.0008	82.94 ±0.53	0.00634
0101	92.96 ±0.67	N/A	83.56 ±0.58	N/A
0110	88.71 ±0.36	0.00064	81.09 ±0.50	0.00064
0111	92.68 ±0.7	0.17384	83.42 ±0.91	0.9124
1000	92.14±0.54	0.00758	82.78 ±0.75	0.00452
1001	92.17 ±0.61	0.00634	83.07 ±0.69	0.08914
1010	92.81 ±0.32	0.36282	81.48 ±0.91	0.00064
1011	92.99 ±0.58	0.9124	83.15 ±0.60	0.03078
1100	92.6 ±0.5	0.14706	82.84 ±0.27	0.0008
1101	92.73 ±0.53	0.23404	82.74 ±0.67	0.00758
1110	92.5 ±0.71	0.17384	83.18 ±0.21	0.04136
1111	92.8 ±0.6	0.23404	83.29 ±0.58	0.33204

results in term of p-values. The aim is to identify all other combinations that may be statistically relatively similar in term of NCC result as 0101 (see Tab. 6.2). Here we only consider the voxelisation using Gaussian kernels as we already know it provides the most accurate reconstructions. Note that the emphasis is on NCC rather than duration as quantitative imaging (i.e. producing the most accurate reconstruction as possible) is the goal in PET. However, we also apply the Wilcoxon signed-rank test on duration (see Tab. 6.3). The idea is to identify which possible good combination provides accurate results the quickest. The p-value is used to compare the performance of Configuration

Table 6.3: Performance comparison in terms of Duration of all the combinations of mutation operators with all flies as Gaussian kernel for the reconstructions with/without noise in the input data. p-value between each entry with Combination 0101. Possible good combinations in term of NCC for both without noise and with noise are highlighted in pink (see Table 6.2).

	Gaussian without noise		Gaussian with noise	
Combination type	**Duration** (in minutes)	**Wilcoxon** (p-value)	**Duration** (in minutes)	**Wilcoxon** (p-value)
0000	14.53 ±1.88	0.0164	14.33 ±1.35	0.0096
0001	15.67±2.02	0.4593	15.47±2.47	0.5287
0010	14.53 ±1.19	0.05614	15.87 ±1.55	0.89656
0011	18.93 ±6.18	0.37886	17.8 ±2.62	0.20766
0100	15.67 ±1.35	0.29834	17.2 ±5.47	0.81034
0101	16.4 ±2.29	N/A	16.47 ±3.31	N/A
0110	15.4 ±1.64	0.09102	14.93 ±1.71	0.0455
0111	16.87 ±3.14	0.75656	16.00 ±2.42	0.67448
1000	15.4 ±1.76	0.25428	15.73 ±2.31	0.32708
1001	15.67 ±2.66	0.29834	15.73 ±2.52	0.75656
1010	16 ±2.42	0.4965	13.93±2.55	0.02642
1011	17.8 ±2.91	0.13104	15.80 ±2.18	0.75656
1100	16.73 ±3.03	0.77948	16.47 ±0.83	0.50926
1101	16.73 ±1.79	0.6818	17.13 ±1.68	0.29372
1110	16.13 ±2.64	0.75656	17.71±3.18	0.27134
1111	17.87 ±3.38	0.29372	18.47 ±2.97	0.101

0101 with other combinations. The size of the samples is 15. For each entry in the table, if the p-value is less than the alpha level of 0.05, then Configuration 0101 is statistically significantly different, and as it has the highest average NCC, we can conclude that 0101 is better. If p-value is greater than 0.05, then there is little evidence to consider 0101 better as they are not significantly different. As a consequence, combinations with p-values higher than 0.05 should also be considered as good candidates. It includes 0001, 0011, 0111, 1010, 1011, 1100, 1101, 1110 and 1111. When considering the duration, all the configurations highlighted in pink or green for this test case have p-values higher than 0.05. It is therefore impossible to objectively distinguish them.

6.3.2 With noise in the input data

The corresponding experiments have been performed with the noisy input data. We can see in Figures 6.7 and 6.8 that the result again proves that the combination of dual and directed mutation operators (0101) seem to show best results (83.56 ±0.58) in terms of NCC in both configurations. It provides better result than the Dual mutation only (0100), which is 82.94 ±0.53. Again, Configurations 0001 and 1001 are also attractive.

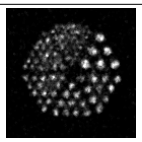
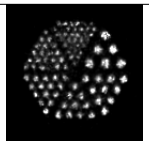
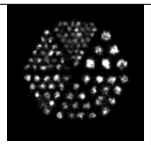
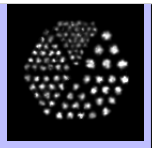
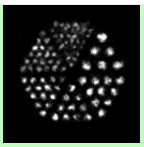
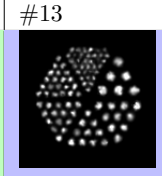
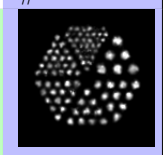
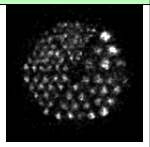
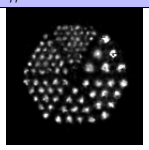
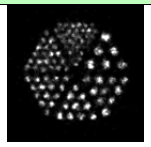
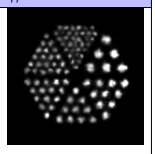
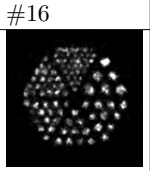
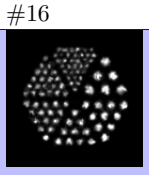
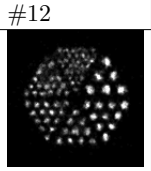
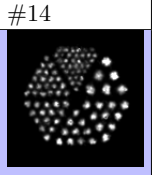
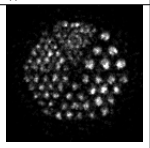
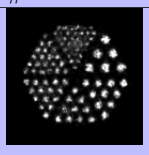
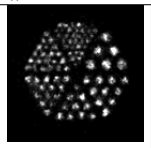
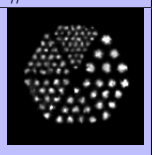
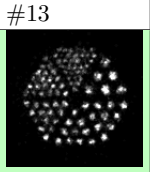
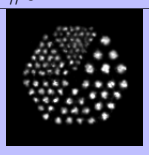
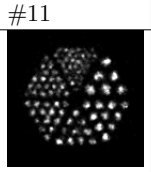
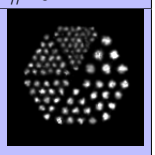
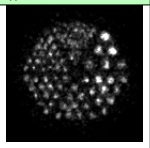
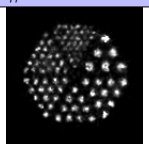
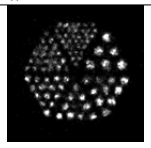
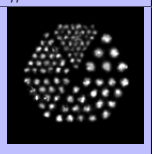
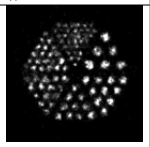
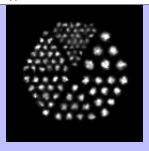
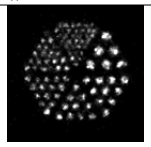
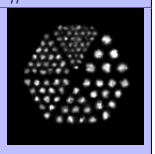
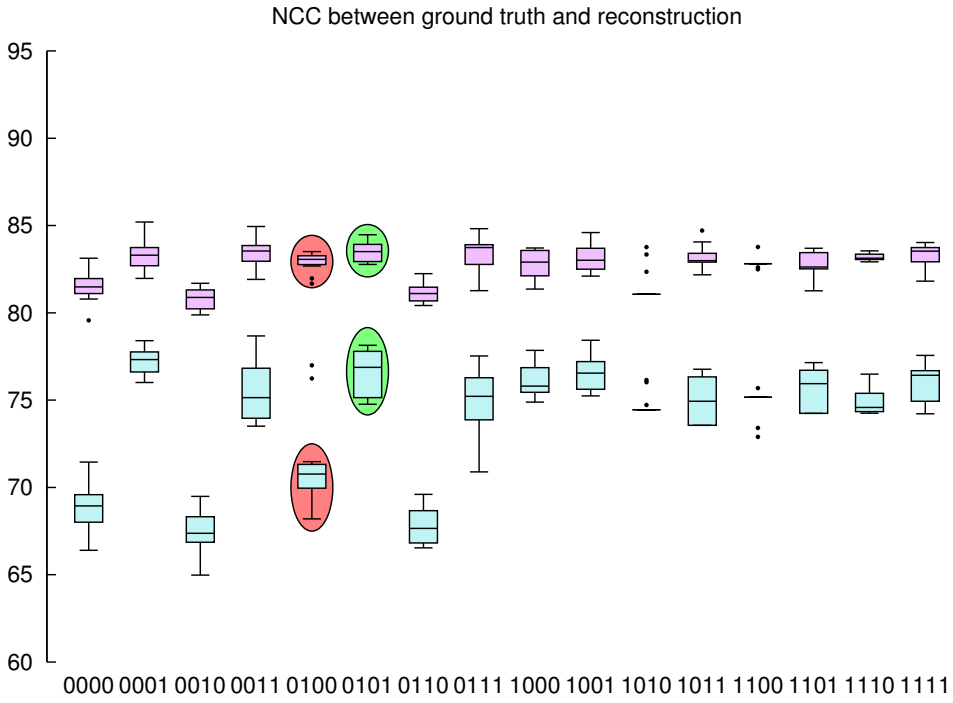
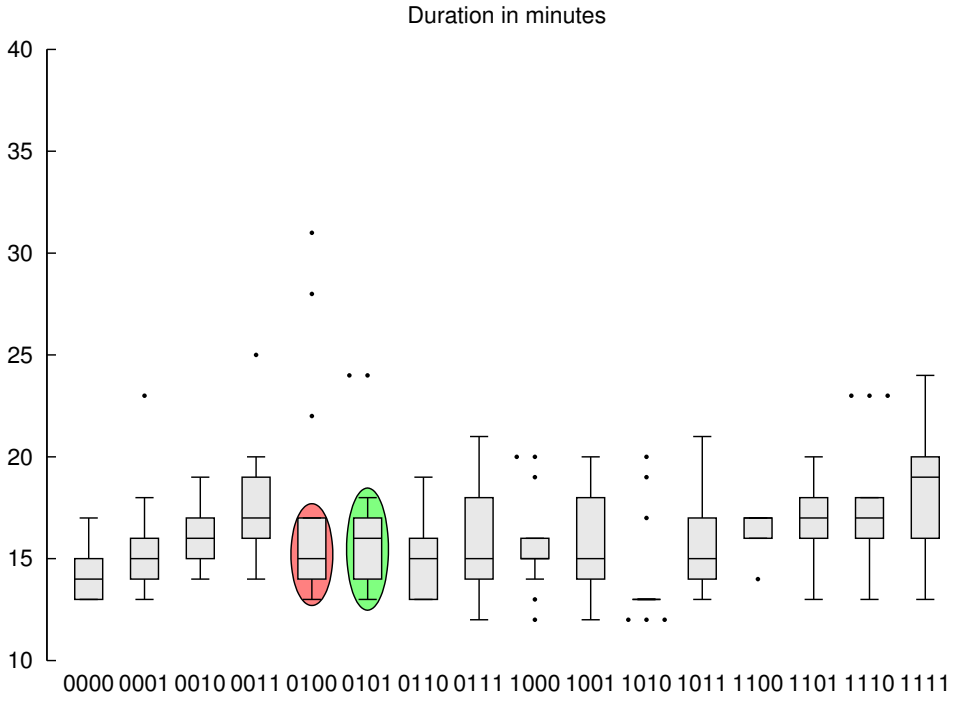
Type/mm #rank	Dirac #rank	Gaussian #rank	Type/mm #rank	Dirac #rank	Gaussian #rank
0000/14.33 #2	 (68.94%) #14	 (81.53%) #13	1000/15.73 #5	 (76.12%) #4	 (82.78%) #11
0001/15.47 #4	 (77.17%) #1	 (83.30%) #4	1001/15.73 #5	 (76.58%) #3	 (83.07%) #8
0010/15.87 #8	 (67.38%) #16	 (80.75%) #16	1010/13.93 #1	 (74.69%) #12	 (81.48%) #14
0011/17.80 #15	 (75.38%) #7	 (83.45%) #2	1011/15.80 #7	 (75.02%) #8	 (83.15%) #7
0100/17.20 #13	 (71.18%) #13	 (82.94%) #9	1100/16.47 #10	 (74.94%) #11	 (82.84%) #10
0101/16.47 #10	 (76.65%) #2	 (83.56%) #1	1101/17.13 #12	 (75.76%) #6	 (82.74%) #12
0110/14.93 #3	 (67.74%) #15	 (81.09%) #15	1110/17.67 #14	 (75.01%) #9	 (83.18%) #6
0111/16.00 #9	 (74.98%) #10	 (83.43%) #3	1111/18.47 #16	 (76.02%) #5	 (83.29%) #5

Figure 6.7: Performance comparison of the different combinations of mutation operators in the presence of noise in the input data. Highlighted in green and blue-violet are the better halves of combinations for the flies as finite points and as Gaussian kernels respectively.

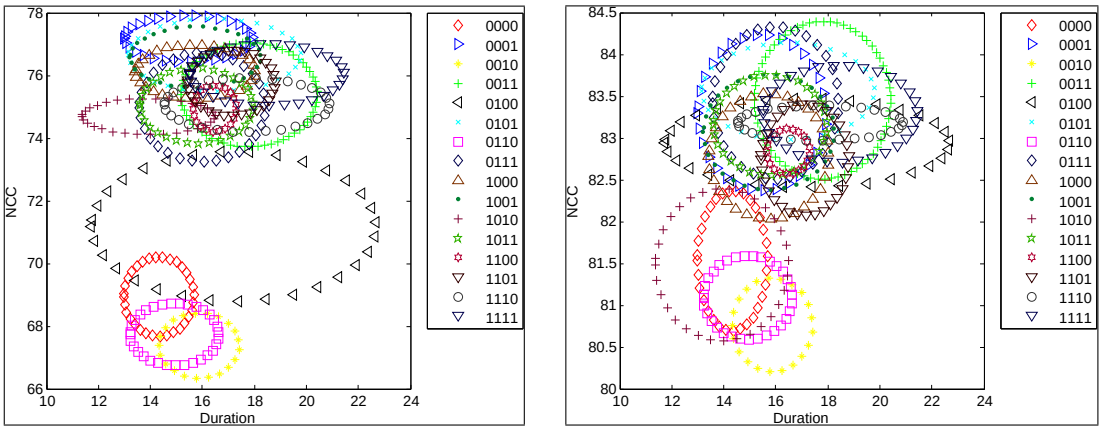


(a) NCC for all flies as a finite point (dark turquoise) and all flies as Gaussian kernel (in purple) [7].



(b) Duration.

Figure 6.8: Performance of mutation operators with noise in the input data. In red is highlighted the performance of our initial implementation with dual mutation only as in [155]. In green is highlighted the performance of the combination of the dual and directed mutation operators.



(a) All the flies **(b)** All the flies with Gaussian kernel

Figure 6.9: Average and standard deviation of NCC and duration (with noise in the input data).

Figures 6.9a and 6.9b show the ellipse visualisation of the results with noisy input data. Results are consistent with our previous observations. The combination of dual and directed mutations (0101) is one of the best in term of NCC (83.56 ±0.58) and computationally acceptable in terms of duration (16.47 minutes ± 3.31).

The Wilcoxon signed-rank test for NCC and duration is also used for this test case (see Tables 6.2 and 6.3 respectively). Again, 0101 has the highest average NCC and it is used as the reference in the test. Only a few configurations have p-values for the NCC that are above 0.05, including 0001, 0011, 0111, 1001, and 1111. Again we cannot conclude that one of these combinations is better in terms of duration.

6.4 Conclusion

The Fly algorithm heavily relies on the mutation operator to find the best positions. In its initial implementation for PET reconstruction by Vidal *et al* [155], only the Dual Mutation operator was used to self-tune the mutation variance. In this chapter we complete the implementation with three other adaptive mutation operators. The probability of the genetic operators are now part of the individuals' genome. It includes a basic mutation operator whose mutation variance is also encoded in the genome. There is also a mutation operator whose variance is a function of the fitness of the individual to mutate. Finally we introduced a new operator, the Directed Mutation, that looks at the history of the individual that is going to be mutated to guide its mutations in

a direction that is likely to be worth exploring based on the experience of the individual's ancestors.

We demonstrate using two test cases (one without noise; one with noise) that this approach and this new operator lead to better results in term of accuracy (improvement of NCC by ~10%) without sacrificing the reconstruction speed. We also studied all the possible combinations of mutation operators to determine if one is better than the others. For all the possible combinations, 15 reconstructions have been performed for each test cases. It is $4^2 \times 2 \times 15 = 480$ reconstructions in total. The Wilcoxon signed-rank test for NCC and duration shows that Combinations 0001, 0101, 0011, 0111, and 1111 perform well in both test cases. For these selection of combinations, the directed mutation is used 5 times, the adaptive mutation 3 times, the dual mutation 3 times, and the basic mutation once. These results show the benefit of our new operators and the benefit to combine several types of mutations in the evolutionary loop.

Further research is needed to evaluate our new operator with other reconstruction data. The operator can also be used in other cooperative co-evolution schemes to improve the cooperation between individuals.

Chapter 7

Digital Arts

The aim of this chapter is to demonstrate how the Fly algorithm can be used in different applications such as digital art. Part of this chapter has been presented at the 20th European Conference on the Applications of Evolutionary Computation, in the track on Evolutionary Computation in Image Analysis, Signal Processing and Pattern Recognition (see Section 1.4) [2].

7.1 Introduction

This chapter is about Evolutionary art such as digital mosaics and painterly rendering. The most common techniques to generate a digital mosaic effect heavily rely on Centroidal Voronoi diagrams. Our method generates artistic images as an optimisation problem without the introduction of any *a priori* knowledge or constraint other than the input image. We adapt the Fly algorithm to produce artistic visual effects from an input image (e.g. a photograph). The primary usage of the Fly algorithm is in computer vision, especially stereo-vision in robotics. It has also been used in image reconstruction for tomography. Until now the individuals correspond to simplistic primitives: Infinitely small 3-D points. In this study, the individuals have a much more complex representation and represent tiles in a mosaic-like image. They have their own position, size, colour, and rotation angle. We take advantage of graphics processing units (GPUs) (see Appendix A) to generate the images using the modern Open Graphics Library Shading Language. Different types of tiles are implemented, some with transparency, to generate different visual effects, such as digital mosaic and spray paint.

To validate our results, a user study has been conducted. It is used to ascertain which version of our algorithm produces the most visually appealing results. We also demonstrate the ability of the algorithm to preserve edges and compared some of our results with similar ones produced with GNU Image Manipulation Program (GIMP) (`http://www.gimp.org/`), an open-source software for image editing.

This chapter is organised as follows. Section 7.2 describes our approach. It explains how the Fly algorithm can be adapted from robotic applications and medical tomography reconstruction into an evolutionary art generator. The penultimate section presents our results. Several images have been generated with different versions of the algorithm. We initially conducted an experiment with 25 participants to judge some these results. It allowed us to determine which version was the most popular amongst the participants. In a subsequent experiment, we ask 41 participants to judge the visual quality of individual images. Some were generated with our approach, others were corresponding images generated with a famous open-source software. Concluding remarks are given in the last section.

7.2 Methodology

7.2.1 Fly algorithm paradigm

We saw in the Section 2.5 there are various approaches to translate an input image into a digital mosaic. The problem can be defined as follows [74]:

> *Given a rectangular region I^2 in the plane $\mathbb{R}^2$, a dataset of N tiles, a set of constraints, and a vector field $\phi(x, y)$ defined on that region, find N sites $P_i(x_i, y_i)$ in I^2 to place the N tiles, one at each site P_i, such that all tiles are disjoint, the area they cover is maximized and the constraints are verified as much as possible.*

In this context, image generation can be studied as a special case of the *set cover problem*, which is NP-complete [87]. It can be solved as an optimisation problem [74,

16]: Find the best set of tiles to generate a rectangular region to approximate a coloured image. Each tile has a colour that represents the specific part of the image it covers. For more realistic reconstruction, the tiles can rotate at a given angle $\phi(x, y)$ following the direction field for that region and may have sightly different sizes. If there are N tiles to place, as each tile has 9 parameters (3-D position, 3 colour components, width, height, and rotation angle), the search space has $9 \times N$ dimensions.

In addition to being a difficult optimisation problem to solve, the digital mosaic generation is also related to topics in computer graphics and visualisation. In particular we saw in Section 2.5 that most of the mosaic synthesis methods are based on Centroidal Voronoi diagrams. In this study, we propose to solve such a problem without the use of any Voronoi diagram. We also consider painterly rendering. Instead we rely on an unsupervised EA based on cooperative co-evolution algorithm principles. The approach we follow is called "Parisian evolution".

7.2.2 Evolutionary image reconstruction

The individuals correspond to extremely simple primitives: The flies. To date, the Fly algorithm has been used to find 3-D positions only. In this study we propose to give flies a finite size so that they now correspond to rectangular tiles. This is because hand-crafted mosaics tend to use such a shape and also because paint brush strokes could be represented using patterned rectangles. Each fly is a vector of 9 elements (see Figure 7.1):

Position is a 3-D point with coordinates (x, y, z), which are randomly generated between 0 and $width - 1$, 0 and $height - 1$, and 1 and -1 respectively (with $width$ and $height$ the number of pixels in the image along the x- and y-axes). An example of an image generated by an initial population is shown in Figure 7.2.

Colour has three components (r, g, b) (for red, green and blue), which are randomly generated between 0 and 1. This is to ensure diversity at the start of the optimisation. Tile colours are evolved rather than assigned deterministically. It leads to better results in term of sharpness when tiles are located at edges between

different regions of the image (see the difference between Figures 7.13a and 7.13b, when tile colours are evolved, and Figures 7.13c and 7.13d, when the tile colours are not evolved).

Rotation Angle is randomly generated between 0 and 360.

Scaling factor has two components (w, h), which control the size of the tile along its horizontal and vertical axes.

Local fitness measures its marginal contribution toward the global solution.

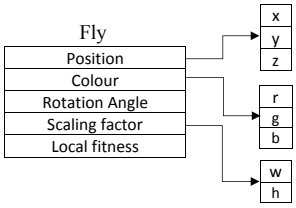


Figure 7.1: Structure of the fly data.

The initial scaling factors are set to make sure the tiles could cover the totality of the image [74]. If there are N individuals, the scaling factors are:

$$d = 0.8\sqrt{(width \times height)/N} \tag{7.1}$$

Due to the randomness in the initial tile positions, tiles overlap. It creates holes in Figure 7.2. We also allow tiles to have different sizes by modifying Eq. 7.1 and adding random variations using the scaling factors w and h. Holes progressively disappear during the evolution process, which aims to optimise the 9 parameters (position, colour, scale, and rotation) of all the N individuals. To achieve this, the algorithm minimises

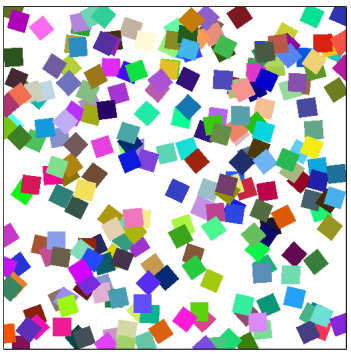


Figure 7.2: Random initial population.

the global fitness function. To assess how good the population is, we compare the input image (ref) with the image generated using the tiles corresponding to the population (pop). We use the sum of absolute errors (SAE) (also known as Manhattan distance) to quantify the error between ref and the computed image pop:

$$\mathrm{SAE}(pop, ref) = \sum_i \sum_j |ref(i,j) - pop(i,j)| \tag{7.2}$$

We favour this metrics as it is fast to compute. Overlapping tiles are not taken into account in the fitness function during the evolutionary process to reduce computational costs.

To improve the population's performance, we need a large proportion of good individuals. The performance of a single fly is evaluated using the local fitness function, which is used during the selection process. In our context, the local fitness is called "marginal fitness", F_m) (see Eq. 7.3). It measures the impact of the selected fly on the global performance of the population. To measure how good or bad the contribution of Fly i is, we use the SAE metrics with the leave-one-out cross-validation method:

$$F_m(i) = \mathrm{SAE}(pop \setminus \{i\}, ref) - \mathrm{SAE}(pop, ref)) \tag{7.3}$$

with $pop \setminus \{i\}$ the image computed with all individuals but Fly i. The numerical value of $F_m(i)$ can be easily interpreted by looking at its sign:

- If the error is greater with Fly i than without, $\mathrm{sgn}(F_m(i)) < 0$, then Fly i damages the performance of the population.
- If the error is smaller with Fly i than without, $\mathrm{sgn}(F_m(i)) > 0$, then Fly i has a positive impact on the performance of the population.
- If the error is the same, $\mathrm{sgn}(F_m(i)) = 0$, then Fly i is not beneficial nor detrimental. It may happen when Fly i is covering similar flies.

We use this principle in our *Threshold selection* operator [161]. To find a fly to kill, pick a random number i between 0 and $N - 1$. If $F_m(i) \leq 0$, then Fly i can be killed, if not pick another random number i until $F_m(i) \leq 0$. To find a fly to reproduce, find one

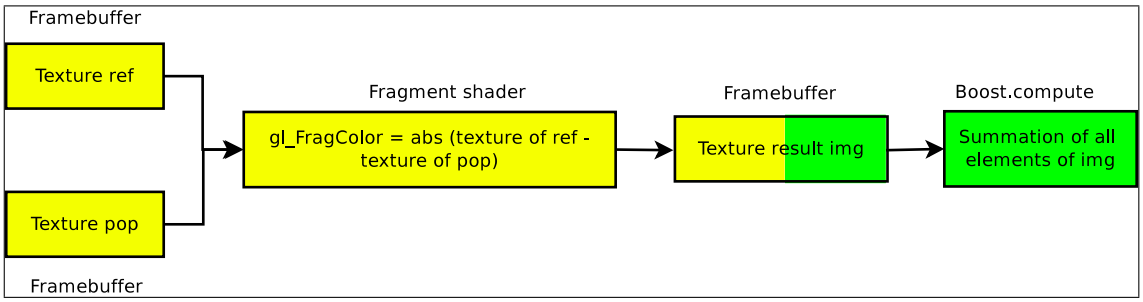


Figure 7.3: Computation of the marginal fitness on GPU for two images using GLSL shaders and Open Computing Language (OpenCL), (see the yellow and green respectively).

whose fitness $F_m(i)$ is strictly positive. During the evolution process, the number of flies whose fitness is negative or null will decrease. There will be more and more good flies; and fewer and fewer bad flies.

Computing the marginal fitness for the problem considered here is time consuming on a central processing unit (CPU). Therefore, all the computations to generate images are performed on a GPU using the GLSL (see Figure 7.3). The image is stored in a 2-D texture using a Framebuffer (FBO). A texture is how a N-D image is stored in a GPU using OpenGL (with $1 \leq N \leq 3$). The texture is then passed to a shader program to compute the pixel-wise absolute error between ref and pop (see Section A.3.2). The sum is also performed on the GPU using the OpenCL implementation of the reduction operator in Boost.Compute [109, 62]. It provides the SAE in an effective manner.

Our implementation is based on a steady state evolutionary strategy (see Figure 4.1). At each iteration of the optimisation process, a bad fly is selected for death and replaced with another one. We use a mutation operator to produce a new fly that is slightly different from the selected good fly. The aim is to create a new fly in the vicinity of a good fly. In this way the new fly is likely to be a good one too. Crossover is not used: If we consider two good flies located at the opposite corners of the image, a new fly in between is very likely to be bad.

The algorithm stops when a stopping criterion is met, e.g. maximum number of iterations, when the evolution process does not improve the performance of the whole population, or when the Threshold selection becomes too slow to find bad flies. A restart mechanism is eventually used to further improve the results. It allows the algorithm to leave a local minimum (see Figure 7.9). Figures 7.4 and 7.5 describe the flowchart that has

been implemented. Restart is implemented by re-launching the optimisation process from scratch using the last best population [58]. It is often used to escape a local minima/maxima.

7.3 Results

In this section we evaluate our method using the results obtained with five test images of increasing complexity (see Row 0 in Figure 7.8). Various tests are performed on these images. In Section 7.3.1, we evaluate 12 different schemes on all the images to determine which scheme provides the 'best' results according to 25 participants. Once a scheme has been selected, we can use it in other experiments. In Section 7.3.2 we study the impact of the background colour. In Section 7.3.3 edge preservation is assessed. A user study comparing our evolutionary art with corresponding images produced with GIMP is presented in Section 7.3.4.

7.3.1 Initial Experiments

We consider the image reconstruction with 12 different schemes (see Table 7.1) using:

- Different image sizes: 256×256 or 512×512;
- Different colour quantisations: 60 colours or full RGB colours (i.e. 2^{24});
- Different types of tiles: square with a border (Figure 7.14c), square without a border, set of lines (Figure 7.14b), or flower (Figure 7.14a);
- With or without restart mechanism.

For each test image, evolutionary reconstructions are obtained with these schemes:

- As a feasibility study, the first two images (Star and Yin & Yang) are relatively simple and have a relatively low resolution. 6 schemes of Table 7.1 are used.

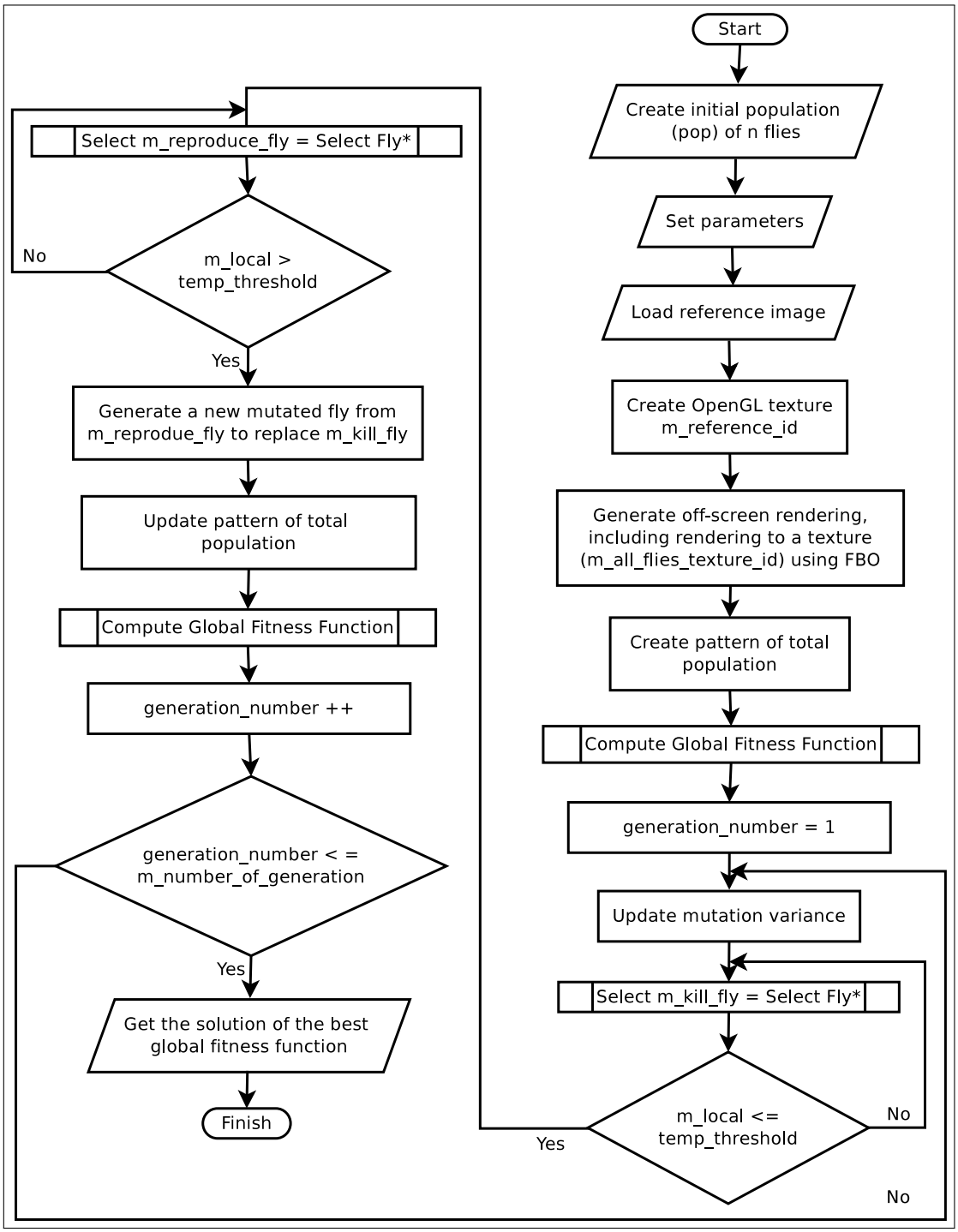


Figure 7.4: Overview of the Fly algorithm for digital art, ($*$ see Figure 7.5a for details on m_local and the select fly function).

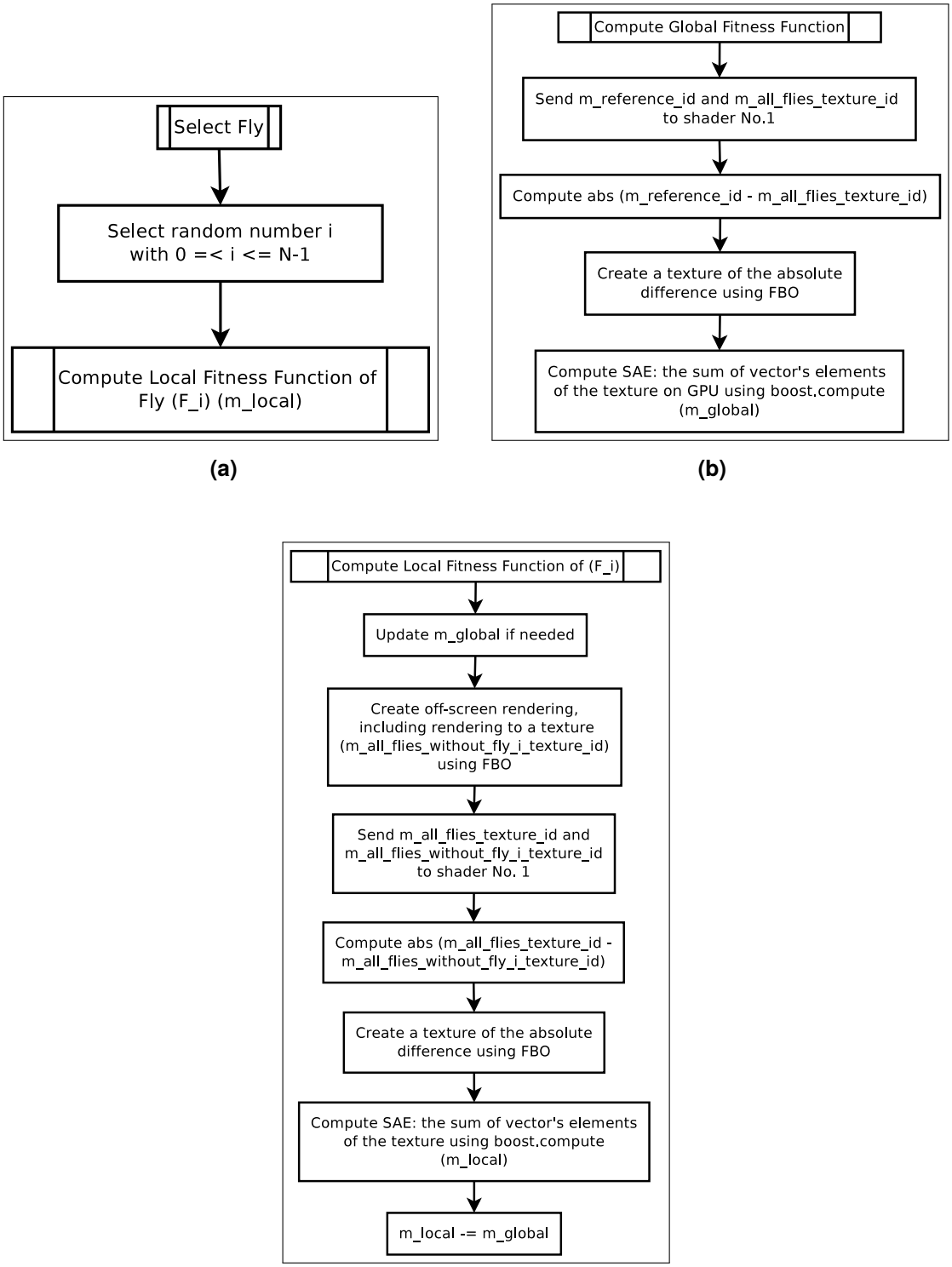


Figure 7.5: Sub-functions of Figure 7.4.

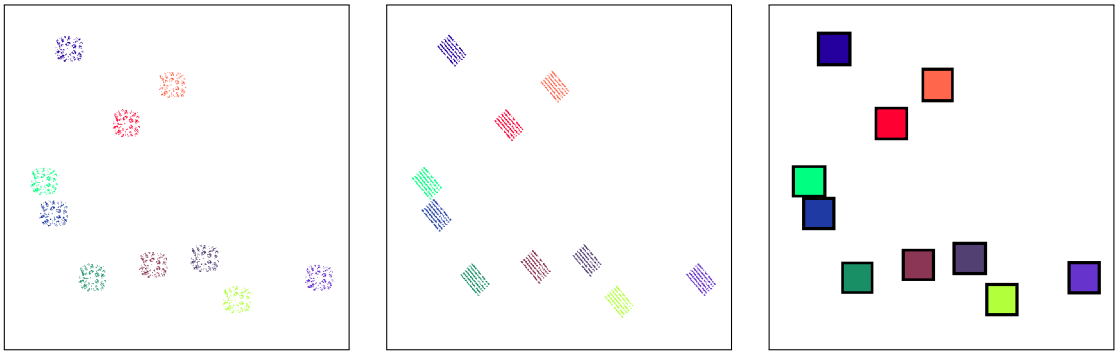
Table 7.1: Summary of all the possible configurations used in Figure 7.8.

#	256×256	512×512	60 colours	Full colour	One shader	Two shaders	Restart
1	✓			✓	✓		
2	✓		✓		✓		
3		✓		✓	✓		
4		✓	✓		✓		
5	✓			✓		✓	
6	✓		✓			✓	
7		✓		✓		✓	
8		✓	✓			✓	
9	✓			✓	✓	✓	✓
10	✓		✓		✓	✓	✓
11		✓		✓	✓	✓	✓
12		✓	✓		✓	✓	✓

- Three other test images are more complex. They are used to evaluate the 12 schemes.

Schemes 1 to 4 use the algorithm presented in Figure 4.1. The flies correspond to uniform rectangles (see "one shader" in Table 7.1). Schemes 5 to 8 use the algorithm presented in Figure 4.1 twice, with one restart between the two runs. The initial population of the first run is random (as in Figure 7.2). The initial population of the second run is the best population of the first run. The flies correspond to complex shapes (see "two shaders" in Table 7.1). Examples of template shapes are given in Figure 7.14. Different shader programs, (see Appendix B), can be used to generate different effects depending on the pixel colour/intensity of the templates. Figure 7.6 shows examples of rendering of the same fly population using different shader programs. Schemes 9 to 12 are almost similar to Schemes 5 to 8. The only difference is that the flies are uniform rectangles during the first run and complex shapes during the second run. The aim is to speed-up computations.

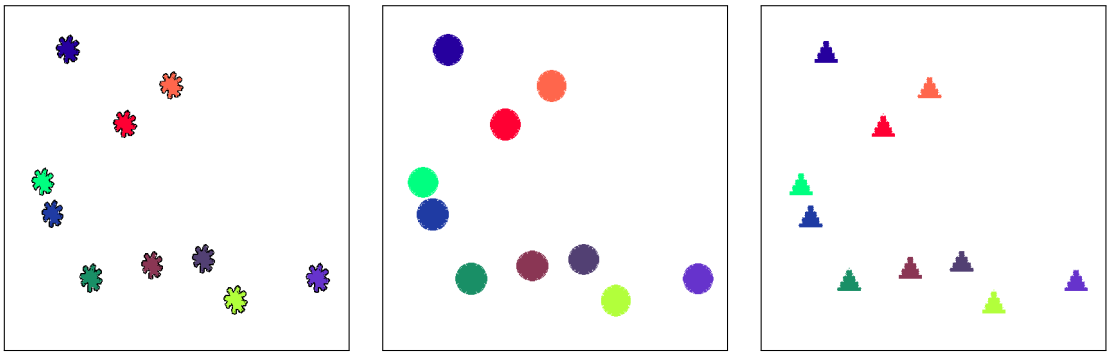
The algorithm is tested with two sets of parameters (see Table 7.2), one with a 22500-D search space, and the other one with a 45000-D search space. We fix the mutation probability to 100% because, as we saw previously, crossover is not suitable in our case. In this implementation, we rely on a mutation rate starting with 0.1 and gradually decrease with the iterations of the algorithm. Figure 7.7 shows the mutation rate for each iteration (in this case the final iteration is 100000). In practice, the parameters chosen by the user are the image (and its size) and the number of individuals. The initial size of tiles is computed depending on the images size and the number of individuals



(a) Mask: Flower (Figure 7.14a).

(b) Mask: Set of lines (Figure 7.14b).

(c) Mask: square (Figure 7.14c).



(d) Mask: Flower (Figure 7.14d).

(e) Mask: Circle (Figure 7.14e).

(f) Mask: Triangle (Figure 7.14f).

Figure 7.6: Rendering of the same flies using different masks and shader programs.

(see Eq. 7.1). Note that the sizes will then undergo evolution. A constraint can be set to restrict their scale to a given interval. We empirically estimated suitable population sizes for different image resolutions. We balanced the number of flies and image size to avoid a premature convergence that would slow down the entire process.

Table 7.2: Parameters used to generate the images in Figure 7.8.

Image size	256×256	512×512
Number of flies	2500	5000
Number of generations	40000	40000
Probability of mutation	100 %	100 %
Probability of crossover	0.0 %	0.0 %
Corresponding scheme	1, 2, 5, 6, 9 and 10	3, 4, 7, 8, 11 and 12
Number of unknowns	$9 \times 2,500 = 22,500$	$9 \times 5,000 = 45,000$

To assess which scheme from Table 7.1 is the most suitable one in term of visual appeal for the end user, Figure 7.8 was printed on A3 paper and we individually asked 25 participants to indicate which image in each column they prefer. Table 7.3 shows the results in percentages. Strategies 11 and 12 are particularly popular among participants. To a lesser extent, the 10^{th} scheme is also popular, the 3^{rd} and 9^{th} also received some votes. The other schemes did not. Schemes 10, 11, and 12 use two shader programs

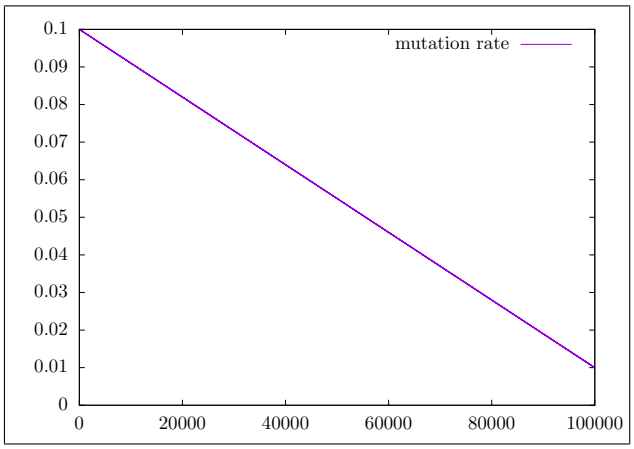


Figure 7.7: Mutation rates.

Table 7.3: Vote results (25 participants voted for their preferred image for each column in Figure 7.8).

Scheme #	Star	Yin Yang	Glasses	Bird	Woman
3	0%	12%	20%	0%	0%
4	0%	0%	0%	0%	4%
9	12%	4%	8%	24%	4%
10	0%	0%	20%	**32%**	0%
11	**88%**	**84%**	**36%**	20%	**48%**
12	0%	0%	16%	24%	44%

and a restart mechanism. Scheme 11 uses full-colour in the original image, Scheme 12 did not.

Figure 7.9 shows the evolution of global fitness with and without restart. It is well known that restart is useful in classical evolutionary algorithms where the solution of the optimisation problem is the best individual of the population. However, very little has been done for the cooperative co-evolution scheme of the Fly algorithm. In [161] a mitosis operator is used to double the size of the population. Here we keep the size of the population constant. When restart is not used, the evolution reaches a plateau then stagnates (see purple curve). After the first restart, the global fitness decreases (see green curve). After the global minimum is found, it is possible that the global fitness increases. A similar phenomenon is observed for the subsequent restarts (see blue, yellow, and orange curves). This experiment demonstrates the benefit of a few restarts in the Fly algorithm.

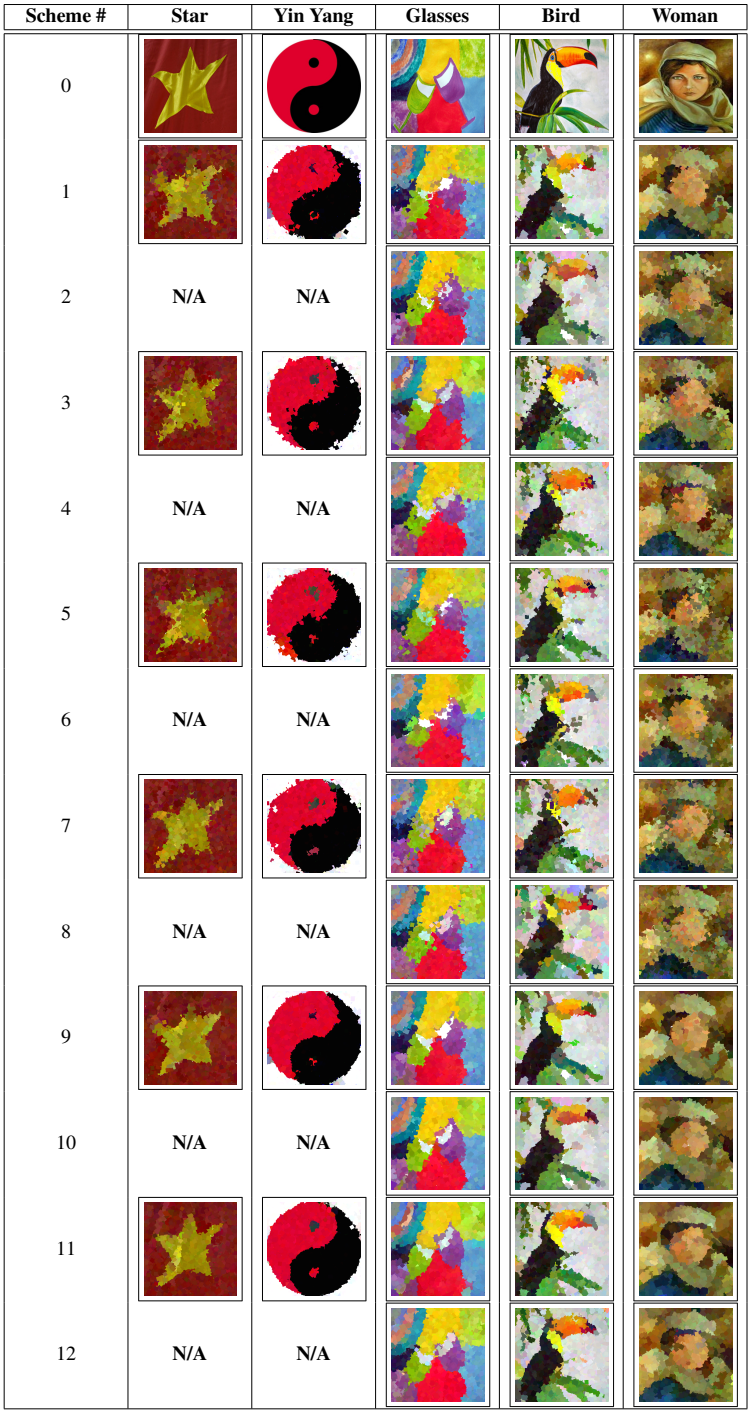


Figure 7.8: Evolutionary art using schemes of Table 7.1. The woman image (Fatima) is from the artist Lubna Ashrafis. Other test images are from the Open Images Dataset (https://github.com/openimages/dataset) under CC BY 4.0 license.

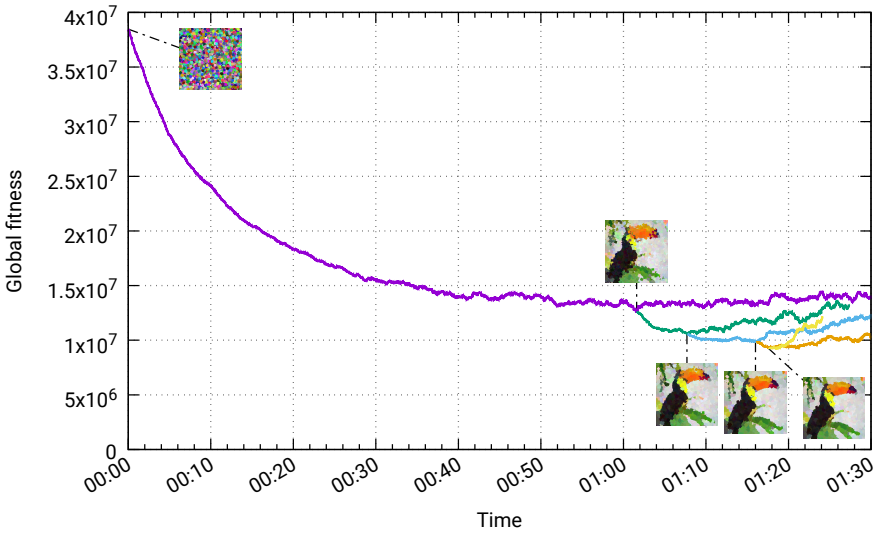


Figure 7.9: Evolution of the global fitness with 4 restarts. Images were computed using a Macbook Laptop with a 2.6 GHz Intel Core i5 CPU with an Intel Iris 5100 GPU.

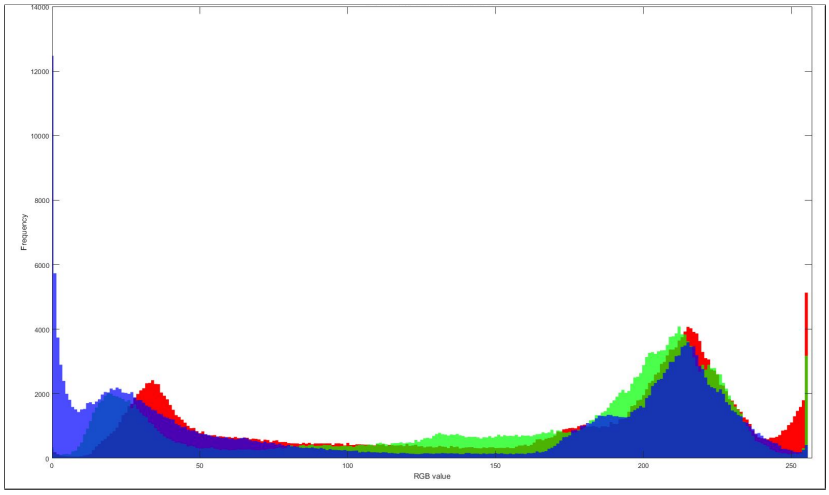


Figure 7.10: The colour histogram in RGB colour space of Bird image.

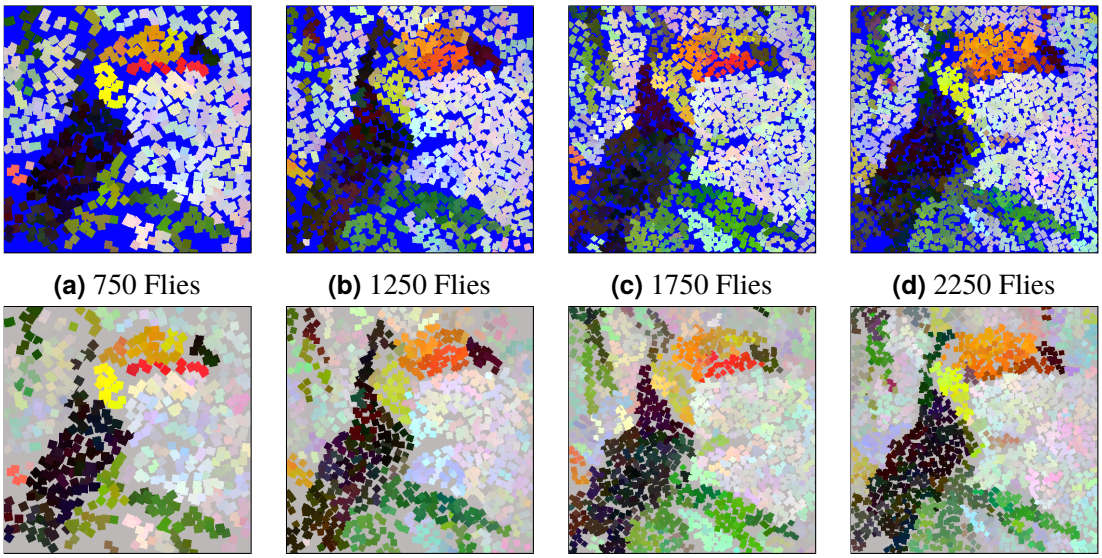
7.3.2 Background Colour

As overlapping tiles and the area coverage are not directly taken into account in the global fitness (Eq. 7.2), it is important to compensate for them in one way or another. It can be done by choosing a background colour that is not present in the original image. For example, the histogram of the Bird image (see Figure 7.10) illustrates that an appropriate background is when we used white (RGB: 255, 255, 255). Figure 7.11 demonstrates that the quality of reconstructed image using white background visually gives more details and similar colours to the reference image comparing with other colour backgrounds. The experiments show that the initial background colour of the reconstructed image has a major impact on the quality of the reconstruction visualisation.

(a) White background **(b)** Red background

(c) Green background **(d)** Blue background

Figure 7.11: Examples of reconstructed Bird image using different background colours.



(a) 750 Flies **(b)** 1250 Flies **(c)** 1750 Flies **(d)** 2250 Flies

(e) Postprocess of (a) **(f)** Postprocess of (b) **(g)** Postprocess of (c) **(h)** Postprocess of (d)

Figure 7.12: Examples of reconstructed Bird image using different number of flies. Top row: reconstructed images with a blue background. Bottom row: replacing the blue background by grey in the reconstructed images.

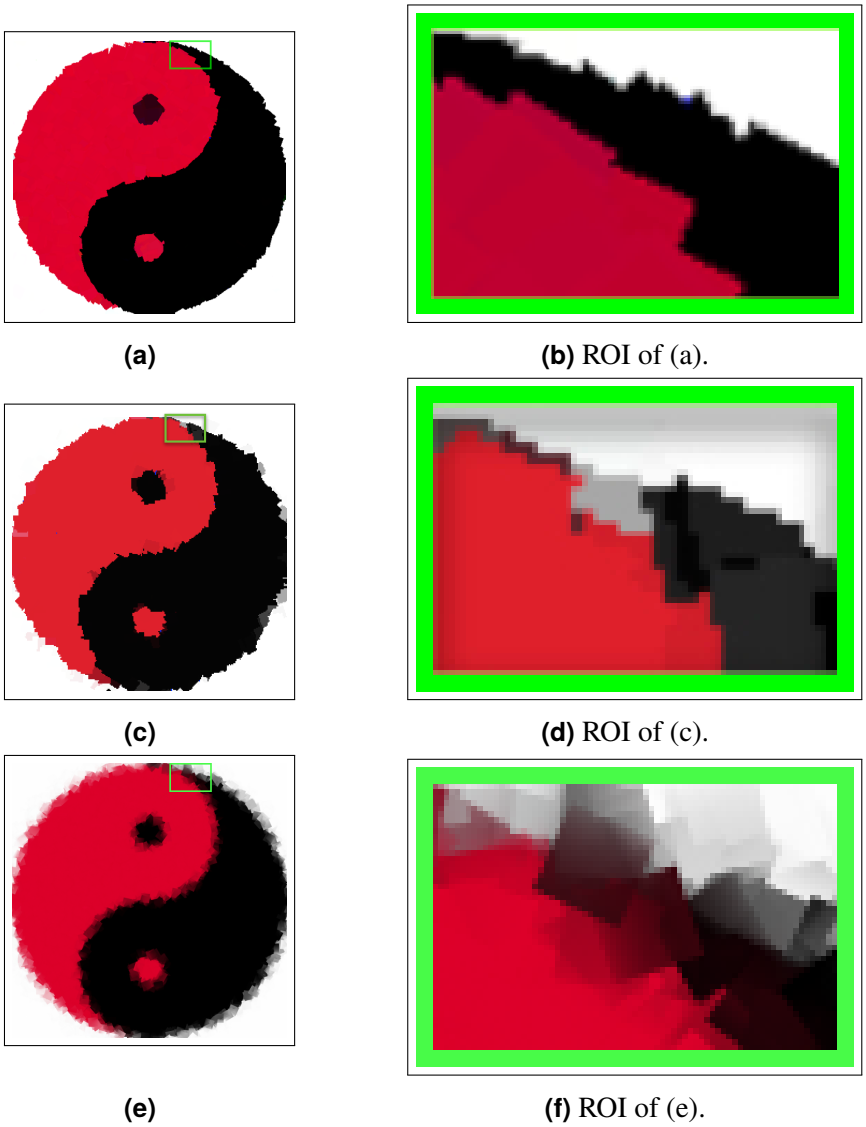
Figure 7.12 shows different experiments with different number of tiles and when a constraint is added on d (Eq. 7.1) and the scaling factors w and h so that the population cannot cover all the image: There will be gaps between tiles. To get a better reconstruction and to widely spread the tiles over the whole image, we choose the blue background as blue (RGB: 0, 0, 255) does not exist in a reference image. As a blue background may not be visually appealing, we can replace corresponding pixels with a more suitable colour.

7.3.3 Edge Preservation

During the generation of the images, tiles can be located at different depth to determine the colour of the closest (visible) tile. It is efficiently implemented using OpenGL's Z-buffer. Depending on the properties of the regions of the image, black tiles may be located behind red tiles, or *vice versa*. The algorithm picks up the discriminated edge between the black and red regions, no matter how small the regions are in the original image compared to the minimum size of a tile. For example, in Figure 7.13b red tiles are over black tiles that are larger than the actual region in the original image. It also shows that our evolutionary algorithm chooses the right rotation angle to follow the curvature of the edges in the original image when colours are evolved. This is not as accurate when colours are picked up directly from the original image as in Figure 7.13d. Edges are even blurrier in images generated using GIMP's filter called GIMPressionist (see Figure 7.13f). This filter implements an Haeberli-like algorithm.

7.3.4 Final User-study

In the following examples, five shapes (or masks) (see Figure 7.14) have been used to generate tiles: Square, flower, stripes, circle and triangle. Figure 7.15 shows the results using the toucan as a test image. The shader program to generate the images can be altered to create different visual effects. The aim is to generate more appealing images. Figures 7.15a and 7.15b have been produced using the same mask (Figure 7.14a) but with slightly different shader programs. Figures 7.15c and 7.15d have been generated using Figure 7.14b as mask, but with a much bigger size and without considering the



(a) **(b)** ROI of (a).

(c) **(d)** ROI of (c).

(e) **(f)** ROI of (e).

Figure 7.13: Edge and depth detection: (a and b) Image reconstructed using our method (evolving colours); (c and d) Image reconstructed using our method (without evolving colours); (e and f) Image reconstructed using GIMPressionist.

rotation angle of the tiles in the second image. Figures 7.15g has been generated using Figure 7.14e as mask. The masks used in Figures 7.15e, 7.15f and 7.15h include an edge. Figure 7.16 shows the process of fitting a tile with certain mask which produce different visual effects by using different fragment shaders. More results are available as videos on YouTube at `http://tinyurl.com/ho5kfvb`.

To demonstrate the usefulness of our approach, further comparison examples have been done between the proposed algorithm and GIMP using two types of mask: Flower and stripe (see Figures 7.15d, 7.15f and 7.18). It can be visually observed that our method leads to better reconstructions in term of edges and colours (brightness). We

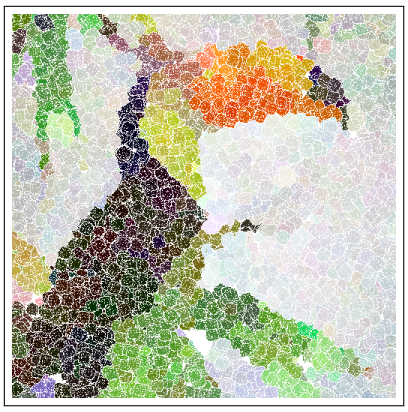
Figure 7.14: Examples of tile templates.

conducted an online study, which is designed using 'Google form' to create the survey sheets. In this study, 18 images (9 created using our method, 9 using GIMP) of different categories, which are Glasses, Bird and Woman (see Figure 7.17), have been used. Each category considers three various masks, which are square, flower and stripes (see Figure 7.14). The images are presented in a random order in the survey. For each image, we ask the participants to rate its quality in terms of visual appearance. 41 participants answered the questionnaire. The results show that our proposed method produces better reconstructed images than GIMP. A Mean opinion score (MOS) metrics has been used to evaluate the results. The rating scale that has been used for evaluating is the Absolute Category Rating scale, which maps ratings between bad and excellent to numbers between 1 and 5, (see table 7.4). The table demonstrates how many times each label was selected for images of our method and for GIMP. For each image, MOS is calculated by Equation 7.4.

$$MOS = \frac{\sum_{n=0}^{N} R_n}{N} \tag{7.4}$$

where R are the individual ratings for a given stimulus by N participants.

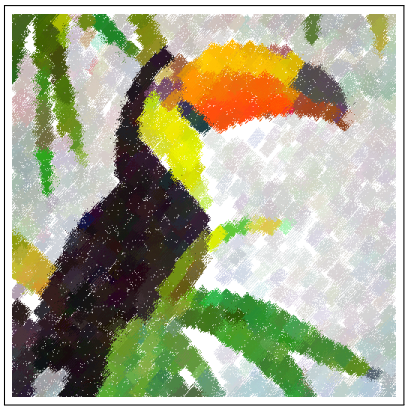
Figure 7.17 shows the MOS for each image of all participants). The MOS values of the images of our proposed method are higher than the MOS of GIMP one. The MOS values are ranked for all images. It shows that the images of our proposed method are mostly in the beginning of the list (1, 2, 3, 4, 6, 8 and 9). While, the images of GIMP are in the end of the list (11, 12, 13, 14, 15, 16, 17 and 18). Except the rank of glasses image of GIMP with the flower mask which is 5. However, the MOS value of that image is 3.1 which is less than the same image of proposed method which is 3.4.



(a) Mask: Flower (Figure 7.14a).

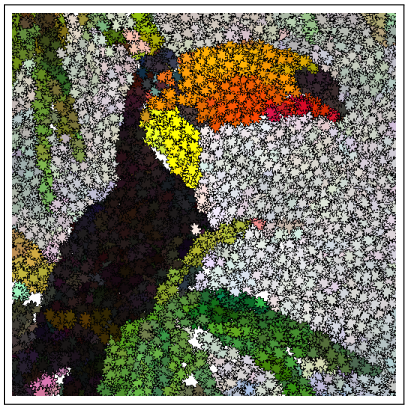
(b) Mask: Flower (Figure 7.14a).

(c) Set of lines (Figure 7.14b).



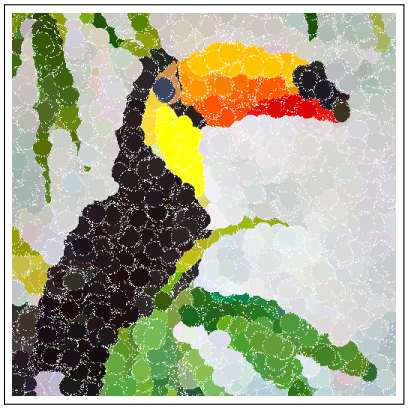
(d) Set of lines (Figure 7.14b).

(e) Mask: square (Figure 7.14c).



(f) Mask: Flower (Figure 7.14d).

(g) Mask: Circle (Figure 7.14e).



(h) Mask: Circle (Figure 7.14e).

Figure 7.15: More appealing visual effects using different masks and shader programs.

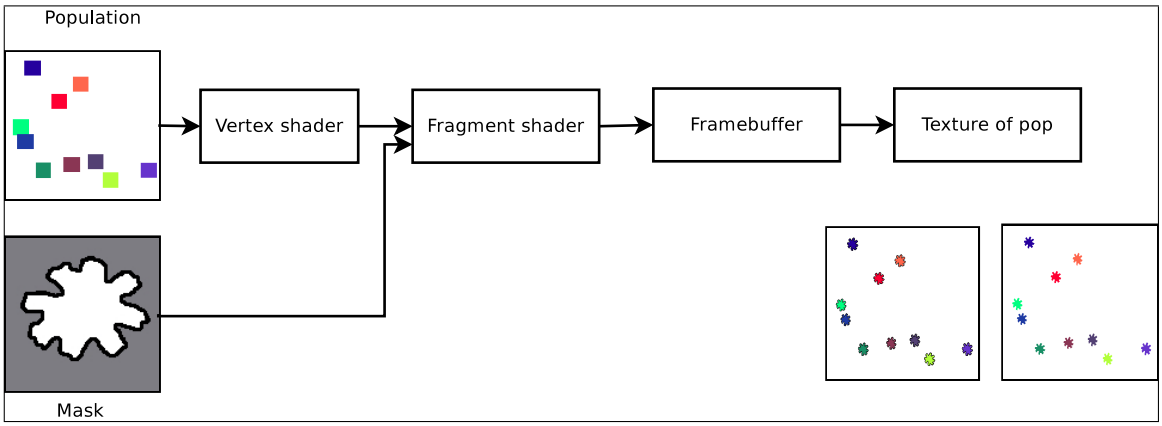


Figure 7.16: Upload different mask using different fragment shaders.

Table 7.4: Absolute category rating of the images of Figure 7.17

Rating	Label	Count of proposed method	Count of GIMP
5	Excellent	32	7
4	Good	86	50
3	Fair	110	78
2	Poor	71	125
1	Bad	19	59

7.4 Conclusion

The problem tackled here lies within the field of Evolutionary art. Our method relies on techniques inherited from CG, AE and scientific computing. We used an AE strategy based on the Fly algorithm for creating visual effects on an image in a fully-automatic fashion. The algorithm optimises the location of tiles in the 3-D space to approximate an input image. We used real-time CG rendering to generate the image data, and GPU computing to calculate the fitness functions. The algorithm can be modified to introduce multiple artistic visual effects. Different templates are used (square, flower, stripes, etc.) to define the shape of tiles.

This chapter has shown that the structure of Flies are more complex with a lot more properties than ever. However, all applications with the Fly algorithm that show the simple notion for Flies as 3-D points (see Chapter 5 as an example). Indeed, complex search space e.g. 45000-D, if not more, has been considered. We manage to significantly enhance the performance of the Fly algorithm. A restarting operator is proposed so as to achieve a better global exploration of the solution space. For more a robust evaluation, an online comparison study (user evaluation survey) has been done, including 18 images

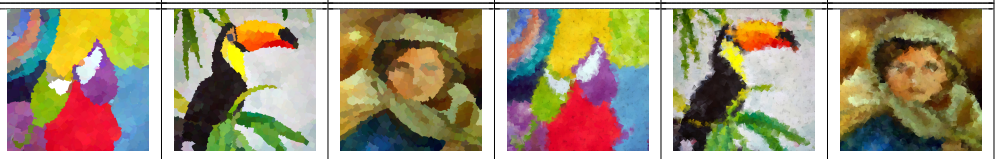
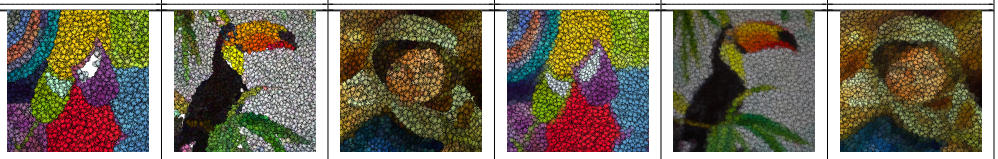
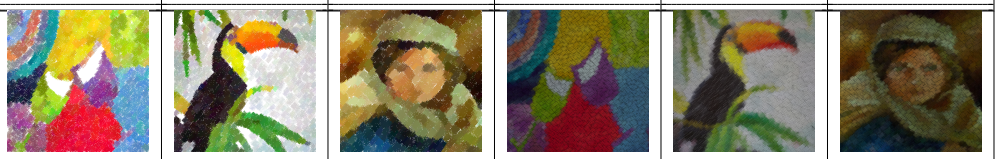
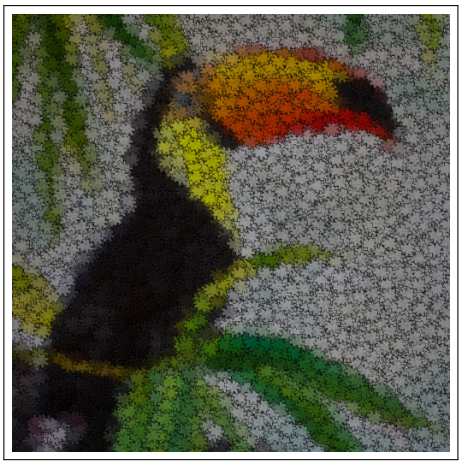
Masks	Images of proposed method			Images of GIMP		
	Glasses	Bird	Woman	Glasses	Bird	Woman
Square						
MOS	2.9	3.3	2.8	2.5	2.6	2.5
Rank	6	4	9	12	11	13
Flower						
MOS	3.4	3.6	2.9	3.1	2.4	2.5
Rank	3	1	6	5	15	14
Stripes						
MOS	2.8	3.4	2.8	2.1	2.0	2.4
Rank	9	2	8	17	18	16

Figure 7.17: Images for online study survey.

(a) Mask: Set of lines.



(b) Mask: Flower.

Figure 7.18: Example of images produced with GIMP's filter (GIMPressionist).

(see Figure 7.17), amongst which 9 were generated by our proposed method and 9 using GIMP. According to the participants, our proposed method produces better images in terms of visual appearance.

Our initial proof-of-concept shows that the Fly algorithm can be used in Evolutionary Arts. Our implementation could be refined to take advantage of more advanced genetic operators, for example to speed-up computations, and to recover fine details. A friendly graphical user interface (GUI) will be added to introduce an optional level of user interaction. It will allow the user to control some parameters of the output image, e.g. shape of tiles, number of tiles, etc. A plugin for an image manipulation program, such as GIMP, will be released to make it available to potential users.

Chapter 8

Conclusions and Future Work

8.1 Introduction

This chapter provides evidence of our attempts to validate our initial hypothesis. It also recaps all the experiments that we have conducted and discusses the most important results. Functionalities and limitations are also identified. This chapter ends by providing recommendations for future work to further improve the Fly algorithm in both medical and digital art applications.

8.2 Overview

Our initial hypothesis was:

> *A relatively overlooked algorithm (Fly algorithm) can provide a competitive alternative (e.g. in term of accuracy, ease of use) to some of the most traditional approaches used in image (2-D or 3-D) reconstruction when considered as an inverse problem.*

To verify this hypothesis, this thesis focused on the Parisian approach (in particular the Fly algorithm) in the field of imaging. While the Fly algorithm has been shown to be promising for solving some complex imaging problem, it still remains relatively unknown as a tool for image processing in the evolutionary computing community. Chapter 3 presented an overview of the principles and applications of EA in image processing. Often AE is used as a blackbox optimisation tool relying on classical

EAs. It is however known as a fact that blackbox optimisation is not the most efficient way to tackle such complex problems. As noted in Chapter 4, only a few studies have investigated the Fly algorithm. It has been demonstrated that it works well in some real-world problems (autonomous robots and tomography reconstruction). In this thesis, we investigated the Fly algorithm by focusing on two main applications: tomographic in nuclear medicine and digital arts. For this purpose, we improved the algorithm (developed new operators and post-processing methods), and we compared its performance to existing techniques. The result have been published in a high ranking journal and in conference proceedings.

In Chapter 7, the first version of evolutionary art using an AE approach based on the Fly algorithm was proposed. In a fully-automatic fashion, our method has benefited from CG and GPU computing to generate visual effects. CG rendering is used to produce the simulated image and GPU computing is used to compute the local and global fitness functions. Initially, the image is computed to approximate an input image by replacing it using a set of discrete square tiles (as in ancient mosaics). This can be considered to be an image reconstruction problem. We then extended our method to produce multiple artistic visual effects by modifying the shape of tiles using different masks (templates), such as a square, a flower, stripes, a circle, a triangle, etc. The structure of the tiles (flies) is more complex and consists of nine parameters, including position, colour, scale and rotation angle. It is important to note that each image is reconstructed either using 2500 or 4500 flies, or even more flies depending on the image size. Therefore, 2500×9 or 4500×9 unknown values must be found. Using classical EAs or any other traditional optimisation method would be computationally too expensive. This is why we propose to solve this problem using the Fly algorithm. The results are compared with images created using a well-known open-source image manipulation software. Our reconstructed images were better in terms of brightness and finding the fine edges between the different areas. The results were evaluated using an online study and a user evaluation survey that included 18 images. The participants gave our images a higher rating in terms of visual appearance.

In Chapter 5, the Fly algorithm is used to solve the complex problem of medical tomographic reconstruction in PET. Here, a structure of individuals (flies) corresponds to a simple primitive (3-D point). The performance of the flies is evaluated using a

"marginal" fitness function. Based on a value of fitness function and using threshold selection, a fly is kept if the fitness is positive. However, a fly is removed if the fitness is negative. The experiment presented in this chapter disproves the hypothesis of our previous implementation in which only flies with a positive fitness were kept. Here, keeping the flies with a negative fitness leads to better the quality results. For more accurate visualisation results, our development investigates the use of implicit modelling to further exploit the point cloud of the final flies step to produce the best quantitative results. The experiment shows that using Gaussian kernels leads to a better reconstructed image than using Metaballs. The spread of Gaussian kernels depends on the local fitness of the flies. Our proposed method is compared with two traditional methods used in nuclear medicine: FBP and OSEM. The experimental results showed that our method provides similar or better reconstructions than OSEM in term of numerical accuracy depending on the input data.

In Chapter 6, four mutation operators are proposed: basic mutation, adaptive mutation variance, dual mutation and directed mutation. The combination of these operators is investigated in order to improve the quality of the reconstructed images in PET. The experiment shows that the configuration of the dual mutation and the directed mutation lead to better results in term of accuracy with an acceptable reconstruction speed time. However, our initial implementation where the dual mutation was used to self-tune the mutation variance.

8.3 Contributions

The work carried out through this research and its results successfully achieved the objectives of this study. Chapters 7, 5 and 6 presented the main contributions of this work for the Fly algorithm in digital arts and tomography reconstruction in nuclear medicine. This includes:

Improving the accuracy of medical imaging (visualisation). The philosophy of applying the initial version of the Fly algorithm has been updated to improve the accuracy of medical imagery. This includes: 1) changing in the concept

in which individuals (flies) are allocated to be in the final population. In the first implementation, the reconstruction solution only relies on positive fitness. However, our updated version proves that the result will be accurate if both positive and negative fitness configurations are included; and 2) a further step has been added to the algorithm to exploit the point cloud using implicit modelling to voxelise the data from a density field using Metaballs and Gaussian kernels. The weight of each fly in the final volume is still 1. However, its local fitness function is considered to be the confidence level in the fly's position. Its footprint is spread over several voxels based on its local fitness function.

Different mutation operators. Four different mutation operators, basic mutation, adaptive mutation variance, dual mutation and directed mutation, are proposed. The operators are fully automated and relatively lightweight in terms of computational maintenance. The operators are investigated in PET reconstruction. The algorithm determines the best configuration, which is the combination of dual and directed mutation that has the most impact on the results in terms of producing accurate results with an acceptable computational processing time.

Proposing a flexible evolutionary art algorithm. The first version of our evolutionary art using the Fly algorithm has been introduced to generate mosaics-like and painterly rendering images. While many previous studies have produced a digital mosaic and painterly rendering using Centroidal Voronoi diagrams, our study uses an algorithm to generate artistic images as an optimisation problem without user intervention so as not to introduce any *a priori* knowledge or constraint other than the input image. Our model takes full advantage of modern GPU. The algorithm relies on a multi-pass algorithm implemented based on FBO to generate image data, GLSL and OpenCL to compute the fitness function. Our algorithm has been extended to generate different visual artistic effects. The algorithm is based on GLSL to update the texture of the flies at different stages of the algorithm. Therefore, different shader programmes are implemented to meet this purpose.

8.4 Limitations

Some limitations exist in this study due to time and resource constraints. Although the Fly algorithm for PET reconstruction can still be improved, it is now mature. Before this study, it was in its infancy. Now we have demonstrated that it can compete with MLEM and OSEM in term of accuracy. The digital art application is at a preliminary stage as a proof-of-concept. First, the reconstruction speed is slow, especially if a reference image contains a lot of details and many colours. The computation time takes about 1-4 hours. However, the speed issue is solved in the tomography application by adding the parallel functionalities to the code to link and execute our implementation with high performance computer (HPC) Wales. Another limitation is that the algorithm is unable to recover fine details of the images. Using a large reference image dimension might solve the issue, but it will also increase the reconstruction time.

8.5 Future work

Different adaptations, tests and experiments still need to be implemented and evaluated. Therefore, some of the ideas and questions that arose during this research still need to be added and answered in the future. Future work will include:

- Clinically, in relation to PET, our method needs to be tested using more realistic data and the use of the latest advances in signal processing such as compressed sensing (also known as compressive sensing, compressive sampling, or sparse sampling) need to be investigated.

- Making it easier to use our method in digital arts by adding a friendly GUI that allows a user to directly interact with the algorithm and control some parameters of the output image, such as the shape of the tiles, the number of tiles, etc.

- A plugin of digital art algorithm for an image manipulation programme, such as GIMP, will be released to make it available to potential users.

- Evaluating our new mutation operator, directed mutation, with other reconstruction data. The operator can also be used in other cooperative co-evolution schemes to improve the cooperation between individuals.

- Our implementation could be refined to take advantage of more advanced genetic operators, for example to speed-up computations, and to recover fine details.

- A hybrid approach should be developed that combines the Fly algorithm with a segmentation method to reconstruct the edges between the distinguished areas with a smaller flies size or to automatically select the region of interests (ROIs) in the reconstructed image to focus the computations on the areas of importance (e.g. those with fine details).

- A progressive resolution approach could be introduce in the digital arts application to decrease the computation time using a "mitosis" operator. Its potential in the tomography implementation (i.e. the reconstruction starts with a small number of flies and the population size increases gradually until the population reaches its maximum size) has been previously demonstrated.

Acronyms

1-D	one-dimensional
2-D	two-dimensional
3-D	three-dimensional
ACO	Ant Colony Optimisation
AE	artificial evolution
AHE	Adaptive Histogram Equalisation
AI	artificial intelligence
API	application programming interface
ART	algebraic reconstruction technique
CAD	Computer-aided diagnosis
CBCT	cone-beam computed tomography
CCEA	cooperative co-evolution algorithm
CG	computer graphics
CMY	cyan-magenta-yellow
CMYK	cyan-magenta-yellow-black
CPU	central processing unit
CT	computed tomography
CUDA	compute unified device architecture
CV	computer vision
CVd	Centroidal Voronoi diagrams
CVT	Centroidal Voronoi Tessellation
DNN	deep neural network

DSC	Dice Similarity Coefficient
EA	evolutionary algorithm
EM	expectation-maximization
ET	emission tomography
EuroGP	Genetic Programming
EVOApplication	European Conference on the Applications of Evolutionary Computation
EvoBAFIN	Natural Computing Methods in Business Analytics and Finance
EvoBIO	Evolutionary Computation, Machine Learning and Data Mining in Computational Biology
EvoCOMNET	Nature-inspired Techniques for Communication Networks and other Parallel and Distributed Systems
EvoCOMPLEX	Evolutionary Algorithms and Complex Systems
EvoCOP	Evolutionary Computation in Combinatorial Optimisation
EvoENERGY	Evolutionary Algorithms in Energy Applications
EvoGAMES	Bio-inspired Algorithms in Games
EvoIASP	Evolutionary Computation in Image Analysis, Signal Processing and Pattern Recognition
EvoINDUSTRY	Evolutionary and Bio-Inspired Computational Techniques within Real-World Industrial and Commercial Environments
EvoKNOW	Knowledge Incorporation in Evolutionary Computation
EvoMUSART	Computational Intelligence in Music, Sound, Art and Design
EvoNUM	Bio-inspired algorithms for continuous parameter optimisation
EvoPAR	Parallel Architectures and Distributed Infrastructures

EvoROBOT	Evolutionary Robotics
EvoSET	Nature-inspired algorithms in Software Engineering and Testing
EvoSTOC	Evolutionary Algorithms in Stochastic and Dynamic Environments
FBO	Framebuffer
FBP	filtered back-projection
FCM	fuzzy C-means clustering
GA	genetic algorithm
GHE	Global Histogram Equalisation
GIMP	GNU Image Manipulation Program
GLGPU	general-purpose computing on graphics processing units
GLSL	Open Graphics Library Shading Language
GLU	OpenGL utility library
GLUT	OpenGL utility toolkit
GPU	graphic processing unit
GUI	graphical user interface
HE	Histogram Equalisation
HLSL	high level shader language
HPC	high performance computer
HSI	hue, saturation, intensity
HVS	human visual system
IE	image enhancement
JIM	jigsaw image mosaics

LB	lower bound
LHE	Local Histogram Equalisation
MAE	Mean absolute error
MAP-Elites	multi-dimensional archive of phenotypic elites
MART	multiple algebraic reconstruction technique
MCM	Monte Carlo with minimisation
MIUA	Medical Image Understanding and Analysis
ML	machine learning
MLEM	maximum-likelihood expectation-maximization
MMBEBHE	Minimum Mean Brightness Error Bi- Histogram Equalisation
MNL	modified nonlinear Laplace
MOS	Mean opinion score
MRF	Markov Random Field
MRI	magnetic resonance image
multi-peak GHE	multi-peak Generalised Histogram Equalisation
NCC	normalised cross-correlation
NPR	non-photorealistic rendering
OpenCL	Open Computing Language
OpenGL	Open Graphics Library
OSEM	ordered subset expectation-maximization
PCA	Principal Components Analysis
PCA-mutation	Principal Components Analysis mutation
PET	positron emission tomography
pixel	picture element
PSNR	Peak signal-to-noise ratio
PTX	parallel thread execution

RGB	red-green-blue
RI	range image
ROI	region of interest
SA	simulated annealing
SAE	sum of absolute errors
SART	simultaneous algebraic reconstruction technique
SIMD	single instruction multiple data
SIRT	simultaneous iterative reconstruction technique
SLAM	simultaneous localisation and mapping
SNR	signal-to-noise ratio
SPECT	single-photon emission computed tomography
SSD	sum of squared differences
SSGA	steady-state genetic algorithm
STDEV	standard deviation
TSP	travelling salesman problem
UB	upper bound
US	ultrasound
voxel	volume element
WICED	Workshop on Intelligent Cinematography and Editing

References

[1] https://www.eg.org/index.php/events/working-groups-events/107-wge/378-intelligent-cinematography-and-editing, Accessed: 2017-03-07 (p. 18).

[2] Z. A. Abbood, O. Amlal and F. P. Vidal, 'Evolutionary art using the fly algorithm', in *Applications of Evolutionary Computation: 20th European Conference, EvoApplications 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings, Part I*, G. Squillero and K. Sim, Eds. Cham: Springer International Publishing, 2017, pp. 455–470, ISBN: 978-3-319-55849-3. DOI: 10.1007/978-3-319-55849-3_30 (p. 119).

[3] R. R. Ahirwal, R. N. Pathak and Y. Jain, 'Contrast enhancement of HDR images using genetic algorithm with efficient fitness value', *International Journal of Computer Science Issues*, vol. 10, no. 6, pp. 70–79, 2013 (p. 42).

[4] T. Ahonen, A. Hadid and M. Pietikäinen, 'Face description with local binary patterns: Application to face recognition', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 2037–2041, 2006, ISSN: 01628828. DOI: 10.1109/TPAMI.2006.244 (p. 18).

[5] F. Alabsi and R. Naoum, 'Comparison of selection methods and crossover operations using steady state genetic based intrusion detection system', *Journal of Emerging Trends in Computing and Information Sciences*, vol. 3, no. 7, pp. 1053–1058, 2012 (pp. 38, 40).

[6] W. Alan, *3D computer graphics*, Third Edit. Addison-Wesley, 2000, ISBN: 0201398559 (p. 173).

[7] Z. Ali Abbood, J. Lavauzelle, É. Lutton, J.-M. Rocchisani, J. Louchet and F. P. Vidal, 'Voxelisation in the 3-D Fly algorithm for PET', *Swarm and Evolutionary Computation*, 2017, ISSN: 2210-6502. DOI: 10.1016/j.swevo.2017.04.001 (pp. 66, 100, 112, 116).

[8] Z. Ali Abbood, J.-M. Rocchisani and F. P. Vidal, 'Visualisation of PET data in the Fly algorithm', in *Eurographics Workshop Vis Comput Biomed*, 2015, pp. 211–212, ISBN: 978-3-905674-82-8. DOI: 10.2312/vcbm.20151227 (pp. 58, 66).

[9] Alliance for Telecommunications Industry Solutions, *ATIS telecom glossary*, Accessed: 2017-08-11, 2001. [Online]. Available: http://www.atis.org/glossary/definition.aspx?id=2014 (p. 76).

[10] A. Andersen and A. Kak, 'Simultaneous algebraic reconstruction technique (SART): A superior implementation of the ART algorithm', *Ultrasonic Imaging*, vol. 6, no. 1, pp. 81–94, 1984, ISSN: 0161-7346. DOI: 10.1016/0161-7346(84)90008-7 (p. 24).

[11] *Applications of evolutionary computation*. [Online]. Available: http://www.evostar.org/2017/cfp_evoapps.php (visited on 2nd Feb. 2017) (p. 31).

[12] T. Bäck, 'Self-adaptation in genetic algorithms', in *Proc 1st European Conf Artif Life*, MIT Press, 1992, pp. 263–271 (pp. 99, 101).

[13] C. Badea and R. Gordon, 'Experiments with the nonlinear and chaotic behaviour of the multiplicative algebraic reconstruction technique (MART) algorithm for computed tomography', *Physics in Medicine and Biology*, vol. 49, no. 8, p. 1455, 2004. DOI: 10.1088/0031-9155/49/8/006 (p. 24).

[14] T. Baeck, D. B. Fogel and Z. Michalewicz, Eds., *Evolutionary Computation 1: Basic Algorithms and Operators*. Taylor & Francis, 2000, ISBN: 978-0750306645 (p. 30).

[15] N. Baek and H. Lee, 'OpenGL ES 1.1 implementation based on OpenGL', *Springer Science+Business Media*, vol. 57, no. December 2010, pp. 669–685, 2012. DOI: 10.1007/s11042-010-0662-4 (p. 175).

[16] S. Battiato, G. D. Blasi, G. M. Farinella and G. Gallo, 'Digital Mosaic Frameworks - An Overview', *Comput Graph Forum*, vol. 26, no. 4, pp. 794–812, 2007. DOI: 10.1111/j.1467-8659.2007.01021.x (pp. 26, 121).

[17] D. Beasley, D. R. Bull and M. R. R., 'An overview of genetic algorithms: Part 1 , fundamentals 1 introduction 2 basic principles', *University Computing*, vol. 15, no. 2, pp. 58–69, 1993 (pp. 33, 34, 38, 40).

[18] H. G. Beyer and H. P. Schwefel, 'Evolution strategies - a comprehensive introduction', *Nat Comput*, vol. 1, no. 1, pp. 3–52, 2002. DOI: 10.1023/A:1015059928466 (pp. 104, 105).

[19] B. Bhanu, S. Lee and J. Ming, 'Adaptive image segmentation using a genetic algorithm', *IEEE transactions on systems, man, and cybernetics*, vol. 25, no. 12, pp. 1543–1567, 1995. DOI: 10.1109/21.478444 (p. 46).

[20] S. Bhattacharyya, 'A brief survey of color image preprocessing and segmentation techniques', *Journal of Pattern Recognition Research*, vol. 1, pp. 120–129, 2011. DOI: doi:10.13176/11.191 (pp. 43, 46).

[21] N. Bissantz, B. A. Mair and A. Munk, 'A statistical stopping rule for MLEM reconstructions in PET', in *IEEE Nuclear Science Symposium Conference Record*, Oct. 2008, pp. 4198–4200. DOI: 10.1109/NSSMIC.2008.4774207 (p. 25).

[22] G. D. Blasi, G. Gallo, U. Catania, V. A. Doria and M. Petralia, 'Puzzle image mosaic', in *IASTED/VIIP 2005*, 2005, pp. 33–37 (p. 26).

[23] J. F. Blinn, 'A generalization of algebraic surface drawing', *ACM Trans. Graph.*, vol. 1, no. 3, pp. 235–256, Jul. 1982, ISSN: 0730-0301. DOI: 10.1145/357306.357310 (pp. 78, 79, 84).

[24] L. B. Booker, D. E. Goldberg and J. H. Holland, 'Classifier systems and genetic algorithms', *Artificial Intelligence*, vol. 40, no. 1-3, pp. 235–282, Sep. 1989, ISSN: 00043702. DOI: 10.1016/0004-3702(89)90050-7 (p. 33).

[25] M. V. Borkar and B. Kurhade, 'A novel approach for face recognition by using near set theory', *Global Journal of Engineering Science and Researches*, vol. 2, no. July, pp. 94–103, 2015 (p. 18).

[26] A. M. Boumaza and J. Louchet, 'Dynamic flies: Using real-time parisian evolution in robotic', *Applications of Evolutionary Computing (2001), EVOIASP2001. Lecture Notes in Computer Science*, vol. 2037, pp. 288–297, 2001 (pp. 56, 58, 62, 63).

[27] ——, 'Mobile robot sensor fusion using flies', *S. Cagnoni et al. (Eds), Evoworkshops 2003. Lecture Notes in Computer Science*, vol. 2611, pp. 357–367, 2003 (pp. 58, 60–62).

[28] A. Bousquet, J. Louchet and J.-M. Rocchisani, 'Fully three-dimensional tomographic evolutionary reconstruction in nuclear medicine', in *Proceedings of the Evolution Artificielle, 8th International Conference on Artificial Evolution*, ser. EA'07, Tours, France: Springer-Verlag, 2008, pp. 231–242, ISBN: 3-540-79304-6. DOI: 10.1007/978-3-540-79305-2_20 (pp. 60, 64–66, 69, 71).

[29] A. Bucci and B. Pollack J., 'On identifying global optima in cooperative coevolution', in *Proceedings of the 2005 conference on Genetic and evolutionary computation - GECCO '05*, New York, New York, USA: ACM Press, 2005, ISBN: 1595930108. DOI: 10.1145/1068009.1068098 (pp. 56, 57).

[30] S. Cagnoni, E. Lutton and G. Olague, *Genetic and Evolutionary Computation for Image Processing and Analysis*. 410 Park Avenue, 15th Floor, #287 pmb, New York, NY 10022, USA: Hindawi Publishing Corporation, 2007, ISBN: 9789774540011 (pp. 20, 21, 31, 33, 36, 57).

[31] S. Cavuoti, M. Garofalo, M. Brescia, M. Paolillo, a. Pescape', G. Longo and G. Ventre, 'Astrophysical data mining with GPU. a case study: Genetic classification of globular clusters', *New Astronomy*, vol. 26, pp. 12–22, Jan. 2014, ISSN: 13841076. DOI: 10.1016/j.newast.2013.04.004 (pp. 31, 36).

[32] S. Chabrier, C. Rosenberger, B. Emile and H. Laurent, 'Optimization-based image segmentation by genetic algorithms', *EURASIP Journal on Video and Image Processing*, pp. 1–23, 2008, ISSN: 1687-5176. DOI: 10.1155/2008/842029 (p. 46).

[33] W. Chainate, P. Thapatsuwan and P. Pongcharoen, 'A new heuristic for improving the performance of genetic algorithm', *World Academy of Science, Engineering and Technology*, vol. 65000, pp. 217–220, 2007 (pp. 30, 37).

[34] K. Chellapilla, 'Combining mutation operators in evolutionary programming', *IEEE T Evolut Comput*, vol. 2, no. 3, pp. 91–96, 1998. DOI: 10.1109/4235.735431 (p. 99).

[35] S.-D. Chen and A. Ramli, 'Contrast enhancement using recursive mean-separate histogram equalisation for scalable brightness preservation', *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 1301–1309, Nov. 2003, ISSN: 0098-3063. DOI: 10.1109/TCE.2003.1261233 (p. 43).

[36] H. Cheng and X. Shi, 'A simple and effective histogram equalisation approach to image enhancement', *Digital Signal Processing*, vol. 14, no. 2, pp. 158–170, Mar. 2004, ISSN: 10512004. DOI: 10.1016/j.dsp.2003.07.002 (p. 43).

[37] X. Cheng and X. Gong, 'An image segmentation of fuzzy C-means clustering based on the combination of improved ant colony algorithm and genetic algorithm', *2008 International Workshop on Education Technology and Training & 2008 International Workshop on Geoscience and Remote Sensing*, pp. 804–808, Dec. 2008. DOI: 10.1109/ETTandGRS.2008.408 (p. 49).

[38] S.-y. Cho and Z. Chi, 'Genetic evolution processing of data structures for image classification', *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 2, pp. 216–231, Feb. 2005, ISSN: 1041-4347. DOI: 10.1109/TKDE.2005.28 (p. 36).

[39] N. S. H. Chu and C. L. Tai, 'Real-time painting with an expressive virtual Chinese brush', *IEEE Comput Graph*, vol. 24, no. 5, pp. 76–85, 2004, ISSN: 02721716. DOI: 10.1109/MCG.2004.37 (p. 25).

[40] G. Colin de Verdière, 'Introduction au GPGPU, aspects matériels et logiciels', *Comptes Rendus - Mecanique*, vol. 339, no. 2-3, pp. 78–89, 2011, ISSN: 16310721. DOI: 10.1016/j.crme.2010.11.003 (p. 174).

[41] P. Collet, E. Lutton and J. Louchet, 'Issues on the optimisation of evolutionary algorithm code', in *IEEE C Evol Computat*, 2002. DOI: 10.1109/CEC.2002.1004397 (p. 100).

[42] J. Collomosse, 'Evolutionary search for the artistic rendering of photographs', in *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*, J. Romero and P. Machado, Eds., Springer, 2007, pp. 39–62. DOI: 10.1007/978-3-540-72877-1_2 (p. 28).

[43] C. Dai, X. Wu and J. Liu, 'Stereo matching using adaptive genetic algorithm', *IEEE, International Conference on Audio, Language and Image Processing. ICALIP* ., pp. 1225–1228, 2008 (pp. 50, 51, 53).

[44] D. Devaraj and B. Yegnanarayana, 'Genetic-algorithm-based optimal power flow for security enhancement', in *IEE Proceedings - Generation, Transmission and Distribution*, IET, 2005, pp. 899–905. DOI: 10.1049/ip-gtd:20045234 (p. 40).

[45] F. Devinck and L. Spillmann, 'The watercolor effect : Spacing constraints', *Vision Res*, vol. 49, no. 24, pp. 2911–2917, 2009, ISSN: 0042-6989. DOI: 10.1016/j.visres.2009.09.008 (p. 25).

[46] A. Dipanda, S. Woo, F. Marzani and J. Bilbault, '3-D shape reconstruction in an active stereo vision system using genetic algorithms', *Pattern Recognition*, vol. 36, pp. 2143–2159, 2003. DOI: doi.org/10.1016/S0031-3203(03)00049-9 (pp. 52, 53).

[47] Y. Dobashi and H. Johan, 'A method for creating mosaic images using Voronoi diagrams', in *Eurographics 2002*, 2002, pp. 341–348 (p. 26).

[48] P. Du, R. Weber, P. Luszczek, S. Tomov, G. Peterson and J. Dongarra, 'From CUDA to OpenCL: Towards a performance-portable solution for multi-platform GPU programming', *Parallel Computing*, vol. 38, no. 8, pp. 391–407, 2012, ISSN: 01678191. DOI: 10.1016/j.parco.2011.10.002 (p. 174).

[49] E. Dunn, G. Olague and E. Lutton, 'Automated photogrammetric network design using the parisian approach', *Applications of Evolutionary Computing*, pp. 356–365, 2005, ISSN: 03029743. DOI: 10.1007/978-3-540-32003-6_36 (p. 58).

[50] A. E. Eiben, R. Hinterding and Z. Michalewicz, 'Parameter control in evolutionary algorithms', *IEEE T Evolut Comput*, vol. 3, no. 2, pp. 124–141, 1999 (p. 99).

[51] G. Elber and G. Wolberg, 'Rendering traditional mosaics', *The Visual Computer*, vol. 19, no. 1, pp. 67–78, 2003. DOI: 10.1007/s00371-002-0175-x (p. 25).

[52] K. Erlandsson, P. Esser, S.-E. Strand and R. V. Heertum, '3D reconstruction for a multi-ring PET scanner', *1993 IEEE Conference Record Nuclear Science Symposium and Medical Imaging Conference*, pp. 1562–1566, 1993. DOI: 10.1109/NSSMIC.1993.373552 (p. 49).

[53] F. H. Fahey, 'Data acquisition in PET imaging', *Journal of Nuclear Medicine Technology*, vol. 30, no. 2, pp. 39–49, 2002, ISSN: 1535-5675 (p. 24).

[54] J. Fang, A. L. Varbanescu, X. Liao and H. Sips, 'Evaluating vector data type usage in OpenCL kernels', *Concurrency and Computation: Practice and Experience*, vol. 27, no. October 2014, pp. 4586–4602, 2015. DOI: 10.1002/cpe (p. 177).

[55] M. E. Farmer and D. Shugars, 'Application of genetic algorithms for wrapper-based image segmentation and classification', *IEEE Congress on Evolutionary Computation*, pp. 1300–1307, 2006. DOI: 10.1109/CEC.2006.1688459 (p. 46).

[56] G. M. Faustino and L. H. De Figueiredo, 'Simple adaptive mosaic effects', in *Brazilian Symposium of Computer Graphic and Image Processing*, 2005, pp. 315–322, ISBN: 0769523897. DOI: 10.1109/SIBGRAPI.2005.46 (p. 26).

[57] R. Q. Feitosa, G. A. O. P. Costa, T. B. Cazes and B. Feijo, 'A genetic approach for the automatic adaptation of segmentation parameters', *in proc. OBIA06*, 2006 (p. 47).

[58] A. S. Fukunaga, 'Restart scheduling for genetic algorithms', in *Parallel Problem Solving from Nature — PPSN V: 5th International Conference Amsterdam,*

The Netherlands September 27–30, 1998 Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 357–366, ISBN: 978-3-540-49672-4. DOI: 10.1007/BFb0056878 (p. 125).

[59] J. J. Furtado, Z. Cai and L. Xiaobo, 'Digital image processing: Supervised classification using genetic algorithm in Matlab toolbox', *Report and Opinion*, vol. 2, no. 6, pp. 53–61, 2010 (p. 19).

[60] A. Fusiello, U. Castellani, V. Murino and D. Informatica, 'Relaxing symmetric multiple windows stereo using Markov random fields', *Springer, Lecture Notes in Computer Science*, vol. 2134, pp. 91–105, 2001 (p. 51).

[61] A. Gaitanis, G. Kontaxakis, G. Spyrou, G. Panayiotakis and G. Tzanakos, 'PET image reconstruction: A stopping rule for the MLEM algorithm based on properties of the updating coefficients', *Computerized Medical Imaging and Graphics*, vol. 34, no. 2, pp. 131–141, 2010, ISSN: 0895-6111. DOI: 10.1016/j.compmedimag.2009.07.006 (p. 68).

[62] B. Gaster, L. Howes, D. R. Kaeli, P. Mistry and D. Schaa, *Heterogeneous Computing with OpenCL*, 1st. Morgan Kaufmann, 2011, ISBN: 0123877660 (p. 124).

[63] P. Gilbert, 'Iterative methods for the three-dimensional reconstruction of an object from projections', *Journal of Theoretical Biology*, vol. 36, no. 1, pp. 105–117, 1972, ISSN: 0022-5193. DOI: 10.1016/0022-5193(72)90180-4 (p. 24).

[64] Z. Gkoutioudi K. and D. Karatza H., 'Multi-criteria job scheduling in grid using an accelerated genetic algorithm', *Journal of Grid Computing*, vol. 10, no. 2, pp. 311–323, Mar. 2012, ISSN: 1570-7873. DOI: 10.1007/s10723-012-9210-y (pp. 31, 33, 36).

[65] M. Gong and Y.-H. Yang, 'Genetic-based multiresolution color image segmentation', *in proc Vision Interface*, pp. 141–148, 2001 (pp. 35, 48).

[66] M. Gong and Y. Y-H., 'Multi-resolution stereo matching using genetic algorithm', *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*, pp. 21–29, 2001. DOI: 10.1109/SMBV.2001.988759 (pp. 35, 50–53).

[67] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Third Edit. Pearson Education, 2010, p. 976, ISBN: 0-13-234563-3 (pp. 19, 20, 42, 46, 62).

[68] R. Gordon, R. Bender and G. T. Herman, 'Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography',

Journal of Theoretical Biology, vol. 29, no. 3, pp. 471–481, 1970, ISSN: 0022-5193. DOI: 10.1016/0022-5193(70)90109-8 (p. 24).

[69] P. Haeberli, 'Paint by Numbers: Abstract Image Representations', *SIGGRAPH Comput. Graph.*, vol. 24, no. 4, pp. 207–214, Sep. 1990. DOI: 10.1145/97880.97902 (p. 26).

[70] R. Hafezi, A. Keshavarz and V. Moshfegh, 'A new algorithm to stereo correspondence using rank transform and morphology based on genetic algorithm', *World Academy of science, Engineering and Technology*, vol. 6, pp. 1064–1067, 2012 (pp. 49, 50, 52, 53).

[71] K.-P. Han, K.-W. Song, E.-Y. Chung, S.-J. Cho and Y.-H. Ha, 'Stereo matching using genetic algorithm with adaptive chromosomes', *Pattern Recognition*, vol. 34, no. 9, pp. 1729–1740, Sep. 2001, ISSN: 00313203. DOI: 10.1016/S0031-3203(00)00114-X (pp. 30, 33, 34, 36, 50, 51, 53).

[72] N. Hansen, S. D. Müller and P. Koumoutsakos, 'Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)', *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003. DOI: 10.1162/106365603321828970 (p. 106).

[73] S. Hashemi, S. Kiani, N. Noroozi and M. E. Moghaddam, 'An image contrast enhancement method based on genetic algorithm', *Pattern Recognition Letters*, vol. 31, no. 13, pp. 1816–1824, Oct. 2010, ISSN: 01678655. DOI: 10.1016/j.patrec.2009.12.006 (pp. 42–44).

[74] A. Hausner, 'Simulating decorative mosaics', in *Proceedings of SIGGRAPH '01*, 2001, pp. 573–580. DOI: 10.1145/383259.383327 (pp. 27, 120, 122).

[75] S. Hegde, C. Gatzidis and F. Tian, 'Painterly rendering techniques: A state-of-the-art review of current approaches', *Computer Animation and Virtual Worlds*, vol. 24, no. 1, pp. 43–64, 2013, ISSN: 1546-427X. DOI: 10.1002/cav.1435 (p. 28).

[76] K. E. Hoff III, J. Keyser, M. Lin, D. Manocha and T. Culver, 'Fast computation of generalized Voronoi diagrams using graphics hardware', in *Proceedings of SIGGRAPH '99*, 1999, pp. 277–286. DOI: 10.1145/311535.311567 (p. 27).

[77] K. R. Hole, V. S. Gulhane and N. D. Shellokar, 'Application of genetic algorithm for image enhancement and segmentation', *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 2, no. 4, pp. 1342–1346, 2013 (p. 42).

[78] P. Hoseini and M. G. Shayesteh, 'Efficient contrast enhancement of images using hybrid ant colony optimisation, genetic algorithm, and simulated annealing', *Digital Signal Processing*, vol. 23, no. 3, pp. 879–893, May 2013, ISSN: 10512004. DOI: 10.1016/j.dsp.2012.12.011 (pp. 43–45).

[79] HPC Wales, *The Tier-1 Infrastructure at Aberystwyth, Bangor, and Glamorgan*, Accessed: 2017-08-11, 2015. [Online]. Available: https://portal.hpcwales.co.uk/docs/userguide/chapter8.html (p. 87).

[80] W. Hu, Z. Chen, H. Pan, Y. Yu, E. Grinspun and W. Wang, 'Surface Mosaic Synthesis with Irregular Tiles', *IEEE T Vis Comput Gr*, vol. 2626, no. c, pp. 1–13, 2015. DOI: 10.1109/TVCG.2015.2498620 (p. 27).

[81] H. M. Hudson and R. S. Larkin, 'Accelerated image reconstruction using ordered subsets of projection data', *IEEE Transactions on Medical Imaging*, vol. 13, no. 4, pp. 601–609, Dec. 1994, ISSN: 0278-0062. DOI: 10.1109/42.363108 (p. 24).

[82] Intel Corporation, *Intel Xeon processor X5650*, Accessed: 2017-08-11. [Online]. Available: http://ark.intel.com/products/47922/Intel-Xeon-Processor-X5650-12M-Cache-2_66-GHz-6_40-GTs-Intel-QPI (p. 87).

[83] T. Isenberg, 'Visualization and processing of higher order descriptors for multi-valued data', in. Springer, 2015, ch. A Survey of Illustrative Visualization Techniques for Diffusion-Weighted MRI Tractography, pp. 235–256, ISBN: 978-3-319-15090-1. DOI: 10.1007/978-3-319-15090-1_12 (p. 25).

[84] R. Jain, R. Kasturi and B. G. Schunck, *Machine Vision*. MC Graw-Hill, 1995, ISBN: 0-07-032018-7 (p. 50).

[85] D. J. Kadrmas, M. E. Casey, M. Conti, B. W. Jakoby, C. Lois and D. W. Townsend, 'Impact of time-of-flight on PET tumor detection', *Journal of Nuclear Medicine*, vol. 50, no. 8, pp. 1315–1323, 2009. DOI: 10.2967/jnumed.109.063016 (p. 24).

[86] M. Kaplan and E. Cohen, 'Computer Generated Celtic Design', *Proceedings of the 14th Eurographics Workshop on Rendering 2003*, vol. 44, pp. 9–19, 2003. DOI: 10.1145/882404.882406 (p. 25).

[87] R. M. Karp, 'Complexity of computer computations', in. Springer, 1972, ch. Reducibility among Combinatorial Problems, pp. 85–103, ISBN: 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2_9 (p. 120).

[88] D. J. Ketcham, R. W. Lowe and J. W. Weber, 'Image enhancement techniques for cockpit displays', Hughes Aircraft Company, Tech. Rep. AD-AO14 928, Dec. 1974 (p. 43).

[89] M. J. Kilgard, *The OpenGL Utility Toolkit (GLUT) Program- ming Interface: API Version 3*. Silicon Graphics, Inc., 1996 (p. 175).

[90] J. Kim and F. Pellacini, 'Jigsaw image mosaics', *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 657–664, 2002 (p. 28).

[91] A. Kumar, 'Encoding schemes in genetic algorithm', *International Journal of Advanced Research in IT and Engineering*, vol. 2, no. 3, pp. 1–7, 2013 (p. 34).

[92] M. Kumar, M. Husian, N. Upreti and D. Gupta, 'Genetic algorith: Review and application', *International Journal of Information Technology and Knowledge Management*, vol. 2, no. 2, pp. 451–454, 2010 (pp. 33, 36, 42).

[93] Y.-K. Lai, S.-M. Hu and R. R. Martin, 'Surface mosaics', *Visual Comput*, vol. 22, no. 9, pp. 604–611, 2006, ISSN: 1432-2315 (p. 27).

[94] S. Le Grand, A. W. Götz and R. C. Walker, 'SPFP: Speed without compromise - a mixed precision model for GPU accelerated molecular dynamics simulations', *Computer Physics Communications*, vol. 184, no. 2, pp. 374–380, 2013. DOI: 10.1016/j.cpc.2012.09.022 (p. 174).

[95] M. Levoy, 'Efficient ray tracing of volume data', *ACM Trans. Graph.*, vol. 9, no. 3, pp. 245–261, Jul. 1990, ISSN: 0730-0301. DOI: 10.1145/78964.78965 (p. 79).

[96] J. Li, L. Yao, E. Hendriks and J. Z. Wang, 'Rhythmic brushstrokes distinguish van gogh from his contemporaries : Findings via automated brushstroke extraction', *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 34, no. 6, pp. 1159–1176, 2012. DOI: 10.1109/TPAMI.2011.203 (p. 18).

[97] M. M. Li, W. Guo, B. Verma, K. Tickle and J. O'Connor, 'Intelligent methods for solving inverse problems of backscattering spectra with noise: A comparison between neural networks and simulated annealing', *Neural Computing and Applications*, vol. 18, no. 5, pp. 423–430, 2009, ISSN: 09410643. DOI: 10.1007/s00521-008-0219-x (p. 18).

[98] Z. Li, S. Qin, X. Jin, Z. Yu and J. Lin, 'Skeleton-enhanced line drawings for 3D models', *Graphical Models*, vol. 76, no. 6, pp. 620–632, 2014, ISSN: 1524-0703. DOI: 10.1016/j.gmod.2014.07.002 (p. 25).

[99] C.-h. Lim, Y.-s. Yoon and J.-h. Kim, 'Genetic algorithm in mix proportioning of high-performance concrete', *Cement and Concrete Research*, vol. 34, no. August 2003, pp. 409–420, 2004. DOI: 10.1016/j.cemconres.2003.08.018 (p. 40).

[100] C. Lois, B. W. Jakoby, M. J. Long, K. F. Hubner, D. W. Barker, M. E. Casey, M. Conti, V. Y. Panin, D. J. Kadrmas and D. W. Townsend, 'An assessment of the impact of incorporating time-of-flight information into clinical PET/CT imaging', *Journal of Nuclear Medicine*, vol. 51, no. 2, pp. 237–245, 2010. DOI: 10.2967/jnumed.109.068098 (p. 24).

[101] W. E. Lorensen and H. E. Cline, 'Marching cubes: A high resolution 3D surface construction algorithm', *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, Aug. 1987, ISSN: 0097-8930. DOI: 10.1145/37402.37422 (pp. 79, 80).

[102] J. Louchet, 'Stereo analysis using individual evolution strategy', in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 1, 2000, 908–911 vol.1. DOI: 10.1109/ICPR.2000.905580 (pp. 12, 29, 52, 58, 62, 63, 73).

[103] J. Louchet and E. Sapin, 'Flies open a door to SLAM', *In EvoWorkshops. Lecture Note in Computer Science*, vol. 5484, pp. 385–394, 2009. DOI: "10.1007/978-3-642-01129-0_43", (pp. 49, 58, 61–63).

[104] J. Louchet, 'From Hough to Darwin: An individual evolutionary strategy applied to artificial vision', in *Artificial Evolution: 4th European Conference, AE'99, Dunkerque, France, November 3-5, 1999. Selected Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 145–161, ISBN: 978-3-540-44908-9. DOI: 10.1007/10721187_11 (p. 33).

[105] ——, 'Stereo analysis using individual evolution strategy', in *Proc Int C Patt Recog*, vol. 1, 2000, pp. 908–911. DOI: 10.1109/ICPR.2000.905580 (p. 58).

[106] J. Louchet, M. Guyon, M.-J. Lesot and A. Boumaza, 'Dynamic flies: A new pattern recognition tool applied to stereo sequence processing', *Pattern Recognition Letters*, vol. 23, no. 1–3, pp. 335–345, 2002. DOI: 10.1016/S0167-8655(01)00129-5 (pp. 62, 63).

[107] L. Lu, F. Sun, H. Pan and W. Wang, 'Global optimization of Centroidal Voronoi Tessellation with Monte Carlo approach', *Transactions On Visualization And Computer Graphics*, vol. 18, no. 11, pp. 1880–1890, 2012. DOI: 10.1109/TVCG.2012.28 (p. 27).

[108] É. Lutton and J. Lévy Véhel, 'Pointwise regularity of fitness landscapes and the performance of a simple ES', in *IEEE Congress on Evolutionary Computation*, 2006, pp. 16–21. DOI: 10.1109/CEC.2006.1688344 (p. 104).

[109] K. Lutz, *Boost.Compute*, http://boostorg.github.io/compute/, Accessed: 2016-10-26 (p. 124).

[110] J. F. A. Madeira, H. L. Pina and H. C. Rodrigues, 'GA topology optimization using random keys for tree encoding of structures', *Structural and Multidisciplinary Optimization*, vol. 40, pp. 227–240, Feb. 2010, ISSN: 1615-147X. DOI: 10.1007/s00158-008-0353-1 (p. 35).

[111] J. Magalhães-mendes, 'A comparative study of crossover operators for genetic algorithms to solve the job shop scheduling problem', *WSEAS Transactions on Computers*, vol. 12, no. 4, pp. 164–173, 2013 (p. 38).

[112] R. Maini and H. Aggarwal, 'A comprehensive review of image enhancement techniques', *Journal of Computing*, vol. 2, no. 3, pp. 8–13, 2010 (p. 42).

[113] R. Malhotra, N. Singh and Y. Singh, 'Genetic algorithms : Concepts , design for optimisation of process controllers', *Computer and Information Science*, vol. 4, no. 2, pp. 39–54, 2011 (pp. 35, 36, 38, 40).

[114] C. Munteanu and A. Rosa, 'Towards automatic image enhancement using genetic algorithms', *in proc of the Congress on Evolutionary Computation*, vol. 2, pp. 1535–1542, 2000. DOI: 10.1109/CEC.2000.870836 (pp. 43, 44).

[115] R. Naoum and A. AL-Sabbah, 'Color image enhancement using steady state genetic algorithm', *World of Computer Science and Information Technology Journal (WCSIT)*, vol. 2, no. 6, pp. 184–192, 2012 (pp. 43, 44).

[116] A. Nguyen, J. Yosinski and J. Clune, 'Deep neural networks are easily fooled: High confidence predictions for unrecognizable images', in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 427–436. DOI: 10.1109/CVPR.2015.7298640 (p. 28).

[117] J. Nickolls and W. J. Dally, 'The GPU computing Era', *IEEE Computer Society*, vol. 30, pp. 56–69, 2010. DOI: 10.1109/MM.2010.41 (p. 172).

[118] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa and K. Omura, 'Object modeling by distribution function and a method of image generation', *Trans. Inst. Elect. Commun. Eng. Japan J68-D*, vol. 4, pp. 718–725, 1985 (p. 80).

[119] G. Ochoa, 'Setting the mutation rate: Scope and limitations of the 1/L heuristic', in *Proc GECCO'02*, 2002, pp. 495–502, ISBN: 1-55860-878-8 (p. 99).

[120] T. Orlowska-Kowalska and J. Lis, 'Application of evolutionary algorithms with adaptive mutation to the identification of induction motor parameters at standstill', *COMPEL*, vol. 28, no. 6, pp. 1647–1661, 2009. DOI: 10.1108/03321640910999923 (p. 104).

[121] J. Owens and M. Houston, 'GPU computing', *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, 2008, ISSN: 00189219. DOI: 10.1109/JPROC.2008.917757 (pp. 171, 172).

[122] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn and T. J. Purcell, 'A survey of general-purpose computation on graphics hardware', *Computer Graphics Forum*, vol. 26, no. 1, pp. 80–113, 2007, ISSN: 01677055. DOI: 10.1111/j.1467-8659.2007.01012.x (pp. 172–174).

[123] L. Panait, S. Luke and J. F. Harrison, 'Archive-based cooperative coevolutionary algorithms', in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 2006, ISBN: 1595931864 (pp. 56, 57).

[124] S. Pandian and V. Modrák, 'Possibilities, obstacles and challenges of genetic algorithm in manufacturing cell formation', *Advanced Logistic systems*, vol. 3, no. 1, pp. 63–70, 2009 (p. 54).

[125] M. Paulinas and A. Ušinskas, 'A survey of genetic algorithms applications for image enhancement and segmentation', *Information Technology and Control*, vol. 36, no. 3, pp. 278–284, 2007 (pp. 43, 54).

[126] O. Pauplin, J. Louchet, E. Lutton and M. Parent, 'Applying evolutionary optimisation to robot obstacle avoidance', *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 9, no. 6, pp. 622–629, 2005 (pp. 56, 62).

[127] H. Peng, F. Long, Z. Chi and W. Siu, 'A hierarchical distributed genetic algorithm for image segmentation', *in pros.2000 Congress on Evolutionary Computation*, vol. 1, no. 3, pp. 272–276, 2000. DOI: 10.1109/CEC.2000.870306 (p. 48).

[128] M. Pérez-meza and R. Montúfar-chaveznava, 'Partial 3D reconstruction using evolutionary algorithms', *World Academy of science, Engineering and Technology*, vol. 18, no. 1, pp. 1001–1006, 2008 (pp. 50, 62).

[129] G. Pignalberi, R. Cucchiara, L. Cinque and S. Levialdi, 'Tuning range image segmentation by genetic algorithm', *EURASIP Journal on Applied Signal*

Processing, vol. 1, pp. 780–790, 2003. DOI: 10.1155/S1110865703303087 (p. 48).

[130] V. O. Prakash, P. Kumar, M. Hanmandlu and S. Chhabra, 'High dynamic range optimal fuzzy color image enhancement using artificial ant colony system', *Applied Soft Computing Journal*, vol. 12, no. 1, pp. 394–404, 2012, ISSN: 1568-4946. DOI: 10.1016/j.asoc.2011.08.033 (p. 42).

[131] J. Qi and R. M. Leahy, 'Iterative reconstruction techniques in emission computed tomography', *Physics in Medicine and Biology*, vol. 51, no. 15, R541, 2006. DOI: 10.1088/0031-9155/51/15/R01 (p. 24).

[132] V. Ramos and F. Muge, 'Image colour segmentation by genetic algorithms', *CoRR*, vol. abs/cs/0412087, 2004 (p. 46).

[133] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart: Frommann-Holzboog Verlag, 1973 (p. 104).

[134] R. J. Rost, *OpenGL Shading Language*, 2^{nd}. Addison Wesley Professional, 2006, ISBN: 0-321-33489-2 (pp. 171, 176).

[135] H. Saito and M. Mori, 'Application of genetic algorithms to stereo matching of images in pixels', *Pattern Recognition Letters*, vol. 16, pp. 815–821, 1995. DOI: doi.org/10.1016/0167-8655(95)00048-L (pp. 51, 52).

[136] E. Sapin, J. Louchet and E. Lutton, 'The fly algorithm revisited - adaptation to CMOS image sensors', in *IJCCI*, 2009, pp. 224–229 (pp. 56, 57, 59, 62).

[137] D. Scharstein and R. Szeliski, 'A taxonomy and evaluation of dense two-frame stereo correspondence algorithms', *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 7–42, 2002 (p. 52).

[138] H.-P. Schwefel, *Numerical optimization of computer models*. Wiley, 1981 (p. 105).

[139] N. Senthilkumaran and R. Rajesh, 'Edge detection techniques for image segmentation – a survey of soft computing approaches', *International Journal of Recent Trends in Engineering*, vol. 1, no. 2, pp. 250–254, 2009 (p. 46).

[140] L. Shepp and Y. Vardi, 'Maximum likelihood reconstruction for emission tomography', *IEEE Transactions on Medical Imaging*, vol. 1, no. 2, pp. 113–122, Oct. 1982, ISSN: 0278-0062. DOI: 10.1109/TMI.1982.4307558 (p. 24).

[141] F. Y. Shih and Y.-T. Wu, 'Enhancement of image watermark retrieval based on genetic algorithms', *Journal of Visual Communication and Image*

Representation, vol. 16, no. 2, pp. 115–133, Apr. 2005, ISSN: 10473203. DOI: 10.1016/j.jvcir.2004.05.002 (p. 45).

[142] G. M. Singh, M. S. Kohli and M. Diwakar, 'A review of image enhancement techniques in image processing', *HCTL Open Int. J. of Technology Innovations and Research*, vol. 5, pp. 1–13, 2013 (pp. 42, 43).

[143] A. Sitek, R. H. Huesman and G. T. Gullberg, 'Tomographic reconstruction using an adaptive tetrahedral mesh defined by a point cloud', *IEEE Transactions on Medical Imaging*, vol. 25, no. 9, pp. 1172–1179, Sep. 2006. DOI: 10.1109/TMI.2006.879319 (p. 72).

[144] G. Slabaugh, B. Culbertson and T. Malzbender, 'A survey of methods for volumetric scene reconstruction from photographs', *International Workshop on Volume Graphics*, vol. 2, no. 7, pp. 81–100, 2001. DOI: 10.1007/978-3-7091-6756-4_6 (p. 50).

[145] F. Soldado, F. Alexandre and H. Paulino, 'Execution of compound multi-kernel OpenCL computations in multi-CPU/multi-GPU environments', *Concurrency and Computation: Practice and Experience*, vol. 28, no. August 2015, pp. 768–787, 2016. DOI: 10.1002/cpe.3612 (p. 177).

[146] M. Srinivas and L. Patnaik, 'Genetic algorithms: A survey', *Computer*, vol. 27, no. 6, pp. 17–26, Jun. 1994, ISSN: 0018-9162. DOI: 10.1109/2.294849 (pp. 33, 34, 36, 42).

[147] D. G. Stork, 'Computer vision and computer graphics analysis of paintings and drawings : An introduction to the literature', in *Int. Conf. Computer Analysis of Images and Patterns (LNCS 5702)*, Berlin: Springer-Verlag, 2009, pp. 9–24 (p. 12).

[148] S. Surti, J. Scheuermann, G. El Fakhri, M. E. Daube-Witherspoon, R. Lim, N. Abi-Hatem, E. Moussallem, F. Benard, D. Mankoff and J. S. Karp, 'Impact of time-of-flight PET on whole-body oncologic studies: A human observer lesion detection and localization study', *Journal of Nuclear Medicine*, vol. 52, no. 5, pp. 712–719, 2011. DOI: 10.2967/jnumed.110.086678 (p. 24).

[149] T. L. Tan, K. S. Sim, C. P. Tso and a. K. Chong, 'Contrast enhancement of computed tomography images by adaptive histogram equalisation-application for improved ischemic stroke detection', *International Journal of Imaging Systems and Technology*, vol. 22, no. 3, pp. 153–160, Sep. 2012, ISSN: 08999457. DOI: 10.1002/ima.22016 (p. 43).

[150] A. Tonda, E. Lutton and G. Squillero, 'Lamps: A test problem for cooperative coevolution', *Studies in Computational Intelligence*, vol. 387, pp. 101–120, 2011, ISSN: 1860949X. DOI: 10.1007/978-3-642-24094-2_7 (p. 60).

[151] S. E. Umbaugh, *Computer vision and image processing : a practical approach using CVIP tools*. Prentice Hall PTR, 1999, ISBN: 0-13-264599-8 (p. 21).

[152] F. P. Vidal, D. Lazaro-Ponthus, S. Legoupil, J. Louchet, É. Lutton and J. Rocchisani, 'Artificial evolution for 3D PET reconstruction', in *Proceedings of the 9th international conference on Artificial Evolution (EA'09)*, ser. Lecture Notes in Computer Science, vol. 5975, Strasbourg, France: Springer, Heidelberg, Oct. 2009, pp. 37–48. DOI: 10.1007/978-3-642-14156-0_4 (pp. 64, 66, 69).

[153] F. P. Vidal, J. M. Létang, G. Peix and P. Clœtens, 'Investigation of artefact sources in synchrotron microtomography via virtual x-ray imaging', *Nuclear Instruments and Methods in Physics Research B*, vol. 234, no. 3, pp. 333–348, Jun. 2005. DOI: 10.1016/j.nimb.2005.02.003 (p. 23).

[154] F. P. Vidal, J. Louchet, J. Rocchisani and É. Lutton, 'New genetic operators in the Fly algorithm: Application to medical PET image reconstruction', *Lect Notes Comput Sc*, vol. 6024, pp. 292–301, 2010. DOI: 10.1007/978-3-642-12239-2_30 (pp. 64, 66, 69, 73, 89).

[155] F. P. Vidal, É. Lutton, J. Louchet and J. Rocchisani, 'Threshold selection, mitosis and dual mutation in cooperative coevolution: Application to medical 3D tomography', in *International Conference on Parallel Problem Solving From Nature (PPSN'10)*, ser. Lecture Notes in Computer Science, vol. 6238, Krakow, Poland: Springer, Heidelberg, Sep. 2010, pp. 414–423. DOI: 10.1007/978-3-642-15844-5_42 (pp. 49, 54, 60, 64, 66, 69, 72, 89, 100, 105, 111, 112, 116, 117).

[156] F. P. Vidal, Y. L. Pavia, J. Rocchisani, J. Louchet and É. Lutton, 'Artificial evolution strategy for PET reconstruction', in *International Conference on Medical Imaging Using Bio-Inspired and Soft Computing (MIBISOC2013)*, Brussels, Belgium, May 2013, pp. 39–46 (pp. 12, 14, 18, 49, 57, 64).

[157] F. P. Vidal, P. Villard and É. Lutton, 'Tuning of patient specific deformable models using an adaptive evolutionary optimization strategy', *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 10, pp. 2942–2949, 2012 (pp. 30, 54, 55).

[158] ——, ‘Tuning of patient specific deformable models using an adaptive evolutionary optimization strategy’, *IEEE T Bio-Med Eng*, vol. 59, no. 10, pp. 2942–2949, 2012. DOI: 10.1109/TBME.2012.2213251 (pp. 36, 104).

[159] F. P. Vidal, D. Lazaro-Ponthus, S. Legoupil, J. Louchet, É. Lutton and J. M. Rocchisani, ‘Artificial evolution for 3D PET reconstruction’, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5975 LNCS, pp. 37–48, 2010, ISSN: 03029743. DOI: 10.1007/978-3-642-14156-0_4 (pp. 58, 65).

[160] F. P. Vidal and J.-M. Rocchisani, ‘Reconstruction in positron emission tomography’, in *Graphics Processing Unit-Based High Performance Computing in Radiation Therapy*, CRC Press, 2015, ch. 12, pp. 185–208 (p. 58).

[161] F. P. Vidal, J. Louchet, J.-M. Rocchisani and É. Lutton, ‘New genetic operators in the Fly algorithm: Application to medical PET image reconstruction’, in *Applications of Evolutionary Computation: EvoApplicatons 2010*, 2010, pp. 292–301. DOI: 10.1007/978-3-642-12239-2_30 (pp. 49, 123, 130).

[162] C. Vo, L. Panait and S. Luke, ‘Cooperative coevolution and univariate estimation of distribution algorithms’, in *Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms - FOGA ’09*, New York, New York, USA: ACM Press, 2009, pp. 141–150, ISBN: 9781605584140. DOI: 10.1145/1527125.1527144 (p. 56).

[163] R. P. Vushnu and V. M. Bhaskaran, ‘Performance analysis of multi clustered parallel genetic algorithm with gray value’, *American Journal of Applied Sciences*, vol. 9, no. 8, pp. 1268–1272, 2012 (p. 34).

[164] E. L. Walker, ‘Perspectives on fuzzy system in computer vision’, in *Conference of the North American Fuzzy Information Processing Society - NAFIPS*, 1998, pp. 296–300. DOI: 10.1109/NAFIPS.1998.715592 (p. 21).

[165] C. Wang, N. Komodakis and N. Paragios, ‘Markov random field modeling, inference & learning in computer vision & image understanding: A survey’, *Computer Vision and Image Understanding*, vol. 117, pp. 1610–1627, 2013 (p. 21).

[166] D. Whitley, ‘An overview of evolutionary algorithms: Practical issues and common pitfalls’, *Information and Software Technology*, vol. 43, pp. 817–831, 14 2001. DOI: dx.doi.org/10.1016/S0950-5849(01)00188-4 (p. 54).

[167] R. P. Wiegand and M. A. Potter, 'Robustness in cooperative coevolution', in *Proceedings of the 8th annual conference on Genetic and evolutionary computation - GECCO '06*, New York, New York, USA: ACM Press, 2006, pp. 215–224, ISBN: 1595931864. DOI: 10.1145/1143997.1144063 (pp. 56, 57).

[168] S. Woo and A. Dipanda, 'Matching lines and points in an active stereo vision system using genetic algorithms', *IEEE, 2000 International Conference on Image Processing*, vol. 3, pp. 332–335, 2000. DOI: 10.1109/ICIP.2000.899382 (pp. 50, 52, 53).

[169] G. Wyvill, C. McPheeters and B. Wyvill, 'Data structure forsoft objects', *The Visual Computer*, vol. 2, no. 4, pp. 227–234, ISSN: 1432-2315. DOI: 10.1007/BF01900346 (p. 82).

[170] G. Yiqiang, W. Yanbin, J. Zhengshan, W. Jun and Z. Luyan, 'Remote sensing image classification by the chaos genetic algorithm in monitoring land use changes', *Mathematical and Computer Modelling*, vol. 51, no. 11-12, pp. 1408–1416, Jun. 2010, ISSN: 08957177. DOI: 10.1016/j.mcm.2009.10.023 (p. 31).

[171] A. L. Yuille, 'Computer vision needs a core and foundations', *IMAVIS*, vol. 30, no. 8, pp. 469–471, 2012. DOI: 10.1016/j.imavis.2011.12.013 (p. 12).

[172] E. Zanaty and A. Ghiduk, 'A novel approach based on genetic algorithms and region growing for magnetic resonance image (MRI) segmentation', *Computer Science and Information Systems*, vol. 10, no. 3, pp. 1319–1342, 2013, ISSN: 1820-0214. DOI: 10.2298/CSIS120604050Z (pp. 36, 47, 48).

[173] K. Zeng, M. Zhao, C. Xiong and S.-C. Zhu, 'From image parsing to painterly rendering', *ACM Trans. Graph.*, vol. 29, no. 1, 2:1–2:11, Dec. 2009, ISSN: 0730-0301. DOI: 10.1145/1640443.1640445 (p. 28).

[174] H. Zhang, J. E. Fritts and S. A. Goldman, 'Image segmentation evaluation: A survey of unsupervised methods', *Computer Vision and Image Understanding*, vol. 110, no. 2, pp. 260–280, May 2008, ISSN: 10773142. DOI: 10.1016/j.cviu.2007.08.003 (pp. 46, 49).

[175] Z. Zhang, C. Hou and J. Yang, 'A stereo matching algorithm based on genetic algorithm with propagation stratagem', in *International Workshop on Intelligent Systems and Applications, 2009. ISA 2009.*, IEEE, 2009, pp. 1–4. DOI: 10.1109/IWISA.2009.5072678 (p. 53).

[176] P. Zingaretti, G. Tascini and L. Regini, 'Optimising the colour image segmentation', *in proc. VII Convegno dell Associazione Italiana per Intelligenza Artificiale*, 2002 (pp. 46, 49).

Appendix A

GPU

A.1 Introduction

At the present time, the GPU has become an inseparable part of most computer systems, from laptops to super computers [121]. It is increasingly in demand in the computer industry for several reasons. Over the past few years, modern GPUs have become sturdy graphics engines because they are based on a highly parallel architecture, single instruction multiple data (SIMD), which is very well suited for computer graphics [134]. The GPU also offers an environment for applications that need a powerful computational engine [121]. Initially referred to as general-purpose computing on graphics processing units (GLGPU), GPUs were developed when only graphics application programming interfaces (APIs) were available for graphics hardware. APIs provided an interface between a programming language and lower level utilities, such as DirectX and OpenGL. With the introduction of general purpose computing APIs for GPUs (e.g. Nvidia's compute unified device architecture (CUDA), and OpenCL), the term GLGPU was re-branded as "GPU computing", and it is now well established in the HPC world. GPUs are suitable for any application that satisfies the criteria below.

- Large problems that require a complex computational process: The driving force for new developments in GPUs is still the video game industry. The main purpose of any GPU is, therefore, to compute a series of complex images as quickly as possible (millions of pixels per second) using a graphics API such as DirectX or OpenGL. The input data correspond to geometrical objects which are defined using polygon meshes and 2-D or 3-D images (known as textures). Note that the data and the tasks performed by the GPU are well suited for the

SIMD architecture. Therefore, GPUs satisfy the requirement of such complex applications by providing a massive amount of computational power [122, 121].

- Complex problems demanding a substantial level of parallelism: Some real-time applications, e.g. image processing, are well-fitted for parallel computing. These applications require many computing/computational units that can process vertices and fragments at the same time. For example, most images contain millions of pixels that require parallel processing [121, 117].

- Focusing on throughput rather than latency: The GPU cares about the quantity of tasks processed per unit of time through the graphic pipeline rather than the time consumed by processing hundreds to thousands of cycles following thousands of primitives [121].

The evolution of the GPU began with a fixed-function special purpose processor (a configuration graphics processor). At this time, the GPU used floating-point arithmetic to calculate 3-D geometry and vertics. The GPU excelled at 3-D graphics with some limitations. Then, the GPU evolved into a full-fledged programmable processor to produce high-dynamic-range scenes by using floating-point arithmetic to configure pixel lighting and colour values. The modern GPU focuses on programmable aspects by simplifying the programmability of both APIs and hardware [121, 117].

A.2 The Graphics pipeline

The application of rendering high-resolution 3-D scenes requires high computation rates with a high degree of inherent parallelism. So, it is important to build hardware which is consistent with the inherent parallelism in an such application. This permits higher performance in graphics applications than can be obtained with traditional microprocessors [122, 117].

The graphic pipeline is well constructed, so it can achieve a high computation rate through active parallel execution in hardware implementation [122]. Figure A.1 shows the main stages of a graphics pipeline, including the following:

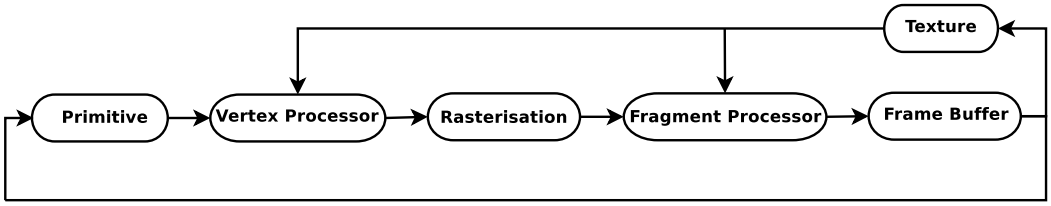


Figure A.1: The modern graphic hardware pipeline

- *Primitive*: In a 3-D coordinate system, the display picture is usually constructed from a list of geometric primitives (which are often triangles). After passing through different steps, these 3-D primitives are coloured and transferred onto the 2-D screen.

- *Vertex Processor*: The geometric primitives are mapped from many vertices that are transferred onto screen and shaded. The vertices are affected, and they interact with the lights of the screen. This stage is heavy because of the independent processing of many vertices.

- *Rasterisation*: This is a method of projecting each triangle onto the screen-space pixel location to generate what is called a "fragment".

- *Fragment Processor*: The final colour of each fragment is determined in this stage using the colour information from the vertices and the texture that is projected onto the surface. Lighting may also be applied at this stage.

At the end of the process, the image is assembled from various fragments with one colour per pixel by keeping the fragment closest to the camera for each pixel location [122].

To speed up the processing, some algorithms may be encoded in the GPU chip. For example, the algorithm used in Z-buffering was first developed by Edwin Catmull in 1974 [6]. The following pseudocode shows the main steps of the Z-buffering algorithm. The basic idea is that the Z-buffer saves its "depth" at each pixel. That is, the algorithm searches for every polygon in the scene, computes the z-value of each pixel on that polygon and saves the smallest z-value corresponding to that pixel. The colour of that pixel is saved in the frame buffer. The Z-buffer prevents a new polygon from being hidden by a previously drawn polygon if the new one has a smaller z-value. The pixels of the new polygon are those closest to the screen.

Algorithm 9 Z-buffering algorithm

```
Init Z-buffer with infinite depth
for all polygon p do
  for all pixel(x, y) ∈ p do
    if p.pixeldepth(x, y) <zbuffer(x, y) then
      zbuffer(x, y) = p.pixeldepth(x, y)
      framebuffer(x, y) = p.pixelcolour(x, y)
    end if
  end for
end for
```

A.3 GPU environments

There are various of programming languages that allow us to run algorithms directly on the graphics vertex and fragment processors using vertex and fragment programs written in a shading language. Example include the C for graphics (Cg) language by Nvidia, DirectX high level shader language (HLSL) by Microsoft and the Open Graphics Library (OpenGL) shading language (GLSL or GLSLang) by the OpenGL Architecture Review Board [122]. Recently, researchers have been trying to balance the load between the development of hardware GPUs and high-level programming interfaces without the need to know about graphics programming. This is to increase the performance of the program execution as well as the flexibility and productivity of the programmer. A variety of programming frameworks have been designed for GPUs, such as parallel thread execution (PTX) assembly code [94], assembly language oriented (CTM, CAL/IL), CUDA [40], OpenCL and DirectCompute [48]. We will focus on the graphic programming frameworks shown below, which are used in this thesis.

A.3.1 OpenGL and the fixed rendering pipeline

OpenGL is a set of functions designed for doing computer graphics. It is not a programming language, so it falls under the category of APIs for computer graphics programming. It is a software interface for the graphics hardware. OpenGL provides two types of rendering: geometric and image primitives. Geometric primitives are points, lines, polygons, spheres, cubes, quadric surfaces, etc. Image primitives are

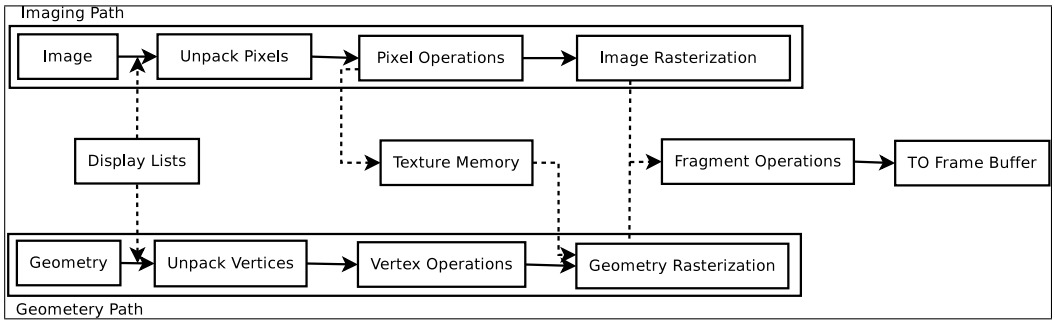


Figure A.2: The OpenGL architecture pipeline

bitmaps and graphics images (e.g. the pixels of an image, which are entered into a program). OpenGL links both types of primitive together using texture mapping. Figure A.2 shows the OpenGL rendering programming pipeline [15].

OpenGL is supplied with environments that produce high-quality scenes. It includes geometric operations (transformation, rotation and scaling), multiple light sources, transparency, blending and the viewing of 3-D scenes using the the concept of a virtual camera.

For the efficient use of OpenGL, two utility libraries have been developed to work alongside with OpenGL. The libraries are OpenGL utility library (GLU) and OpenGL utility toolkit (GLUT).

- GLU is a set of functions that work independently from the OpenGL package. The GLU provides higher-level drawing routines to make the OpenGL more productive. It is used to provide *i*) more complex primitives, such as curves, sphere or cylinder, and *ii*) an easy map between screen and world coordinates.

- GLUT is a set of functions that manage the display window and handle the tools to create GUIs for OpenGL. GLUT helps the user to input an action from the mouse or the keyboard. More details can be found in [89].

A.3.2 GLSL

GLSL is a high-level programming language used only by the OpenGL graphics API. It aims at the implementation of specific algorithms called shader programs. A shader

program is executed directly in the GPU. The program consists of *i*) a vertex shader (vertex program) and *ii*) a fragment shader (fragment program). From this point, we will use the term"shader" because we have implemented algorithms related to "shading" in this thesis. Both types of shader substitute major parts of the vertex or (fragment) operations of the tradtional fixed function of the geometry processing or (rasterisation) unit, respectively [134]. The code in Listings A.1 and A.2 provide examples of the fragment and vertex shader written in GLSL for OpenGL 2.0 to calculate the absolute error between two images (`img_1` and `img_2`).

Listing A.1: Fragment shader

```
#version 120
uniform sampler2D img_1;
uniform sampler2D img_2;
varying vec2 texcoord;

void main()
{
    vec4 texcolor_1 = texture2D(img_1, texcoord);
    vec4 texcolor_2 = texture2D(img_2, texcoord);

    gl_FragColor = abs(texcolor_1 - texcolor_2);
}
```

Listing A.2: Vertex shader

```
#version 120
varying vec2 $texcoord$;

void main()
{
    texcoord = gl_MultiTexCoord0.xy
}
```

A.3.3 OpenCL

OpenCL is an open source used to describe data-parallel kernels and write portable software for a wide range of highly parallel processors that have flexible functionalities working and executing across CPUs and GPUs. OpenCL reduces the complexity of writing a code for GPUs, especially for the the difficulties that occur while adapting general-purpose code to a graphics API. The OpenCL provides easy functions and a wide set of programming APIs, which are based on the C language [145, 54].

A.3.4 Discussion

This chapter has explained the main tools used in this thesis for programming GPUs.

The objectives of using OpenGL are listed below:

- The first aim is to produce an interactive 3-D application. OpenGL provides easy tools to map a 3-D object onto a screen. There are three different matrices, model, view and projection, to move from world coordinate to camera coordinate and then to object coordinate. 1) The model matrix defines the position (x,y,z) coordinates of the primitives, i.e. where an object is drawn on the screen from the site of the object that exists in the world, 2) The view matrix is functionally equivalent to a camera. It defines the position, location and orientation of the camera used for different matrices, i.e. transformation, rotation and scaling. It is used to transform the vertices of a primitive from world to screen coordinate systems. and 3) The projection matrix defines the characteristics of the camera, such as the field of view or the projection method.

- The second aim is to render and visualise our data and results using the highest quality graphics. OpenGL provides facility of flexibility moving across different operating systems, Mac, Linux, and Windows, which are used in this thesis. That makes our implementation portable and executable in most, if not all, recent hardware.

- The third objective is to ensure that scenes are rendered in the same way as they are in the real world in order to use the Z-buffer algorithm. The Z-buffer method tests pixel depth and checks the current position (z-coordinate) against the history of the data stored in that buffer. Thus, the algorithm shows the colour of the last pixel, which expresses the last position of that pixel.

We used GLSL as a high-level shading language because its syntax is similar to the C programming language. It can be directly compiled and run on graphic hardware. This language takes advantage of GPUs, which are considerably parallel processors; this makes our implementation work faster. GLSL provides the opportunity for the programmers to modify pipeline shaders. Hence, we were able to build our own vertex shader and fragment shader.

Open-source OpenCL is used to accelerate mathematical processes and provide a comfortable floating point for implementation, e.g in a digital signal processing application. We used OpenCL in our applications instead of CUDA for the following reasons:

- OpenCL kernels can be used seamlessly in different platforms (CPUs and GPUs) using various programming languages, such as C, C++, Java, Python, JavaScript, Haskell, Perl, Ruby and so on.

- OpenCL supports many varieties of hardware, but CUDA is a closed Nvidia framework. OpenCL makes our application easily portable on recently manufactured hardware.

Appendix B

Shader programs

Listing B.1: Shader program that is suitable for producing a set of stripes or a circle effect.

```
#version 120

varying  vec2 texcoord;
uniform  sampler2D image;
uniform  vec3 fly_colour;
uniform  vec3 fly_concrete_colour;

void main()
{
   vec4 texcolor = texture2D(image, texcoord);
   if (texcolor.r == 1.0)
        gl_FragColor = vec4(fly_colour,1.0);
    else if (texcolor.r > 0.4) and (texcolor.r < 0.4)
       gl_FragColor = vec4(fly_concrete_colour,1.0);
    else
       discard;
}
```

Listing B.2: Shader program that is suitable for producing a fly shape with black edge or a flower shape with black edge effect using mask 7.14d.

```
#version 120

varying  vec2 texcoord;
uniform  sampler2D image;
uniform  vec3 fly_colour;
uniform  vec3 fly_concrete_colour;

void main()
{
   vec4 texcolor = texture2D(image, texcoord);
   if (texcolor.r == 1.0)
        gl_FragColor = vec4(fly_colour,1.0);
    else if (texcolor.r > 0.4) and (texcolor.r < 0.6)
        discard;
    else
        gl_FragColor = vec4(fly_concrete_colour,1.0);
}
```

Listing B.3: Shader program that is suitable for producing a spray painting effect using mask 7.14a.

```
#version 120

varying  vec2 texcoord;
uniform  sampler2D image;
uniform  vec3 fly_colour;
uniform  vec3 fly_concrete_colour;

void main()
{
   vec4 texcolor = texture2D(image, texcoord);
    if (texcolor.r < 0.5)
        discard;
    else
        gl_FragColor = vec4(fly_colour,1.0);
}
```

Listing B.4: Shader program that is suitable for producing a square or a sold flower effect.

```
#version 120

varying  vec2 texcoord;
uniform  sampler2D image;
uniform  vec3 fly_colour;
uniform  vec3 fly_concrete_colour;

const    float offset = 0.5 / 255.0;
void main()
{
   vec4 texcolor = texture2D(image, texcoord);
   if (texcolor.r < offset)
        discard;
    else if (texcolor.r > 0.4) and (texcolor.r < 0.6)
        gl_FragColor = vec4(fly_concrete_colour,1.0);
    else
        gl_FragColor = vec4(fly_colour,1.0);
}
```

Listing B.5: Shader program that is suitable for producing a triangle effect.

```
#version 120

varying  vec2 texcoord;
uniform  sampler2D image;
uniform  vec3 fly_colour;
uniform  vec3 fly_concrete_colour;

void main()
{
   vec4 texcolor = texture2D(image, texcoord);
    if (texcolor.r == 1.0)
        gl_FragColor = vec4(fly_colour,1.0);
    else
        discard;
}
```