ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

DIPLOMOVÁ PRÁCE

Atlas pro separaci medicínských dat

Katedra počítačů
FEL ČVUT Praha

1996 **Pavel Mrázek**

Shape Model for Segmentation of Medical Data

Pavel Mrázek

1996

Abstrakt

Ukazuje se, že vzhledem k povaze dat nejsou porozumění či segmentace dat z lékařské vyšetřovací metody magnetická rezonance (MR) možné bez využití anatomických znalostí. Tato práce se zabývá tím, jak tyto znalosti reprezentovat a jak je využít při hledání anatomických objektů. V textu nejdříve shrneme dřívější přístupy k inteligentnímu zpracování MR dat. V dalších kapitolách představíme podrobně náš model, založený na tvaru a vzhledu anatomického útvaru, který má být v datech nalezen. Znalosti o objektu jsou uloženy ve formě spojených bodů tvořících hranici modelu a k těmto bodům příslušející informací o průběhu stupňů šedi v jejich blízkosti. Detailně vysvětlíme metody uplatňující se v průběhu hledání modelu v obraze, t.j. globální a obecnou optimalizační metodu - genetické algoritmy a lokální techniku úpravy parametrů modelu. Rysy obou metod jsou přiblíženy na praktických příkladech. Model byl navržen pro dvoj- i trojrozměrná data, přičemž nejvýznamnějším rozdílem a rozšířením jsou geometrické transformace ve třech rozměrech.

Abstract

This work concentrates on the domain of medical Magnetic Resonance (MR) data processing. The character of the data makes segmentation and search for objects in an image impossible without exploitation of anatomical knowledge. After a review of earlier approaches to knowledge-based MR data segmentation, we will thoroughly present our model. It is designed to allow representation of information about an anatomical structure and to employ such information during the process of the structure's extraction from MR data. Knowledge on an object's shape and appearance is stored in form of connected points forming the model boundary and grey-level appearance at these model points. Two methods take part when employing the model for data segmentation: global Genetic Algorithms and a local parameter refinement technique. Features of both of them are presented on practical examples. The model has been developed for either two- or three-dimensional cases, while the most important difference and extension are the 3D geometrical transformations.

Slovo českému čtenáři

Stalo se jistou módou na Katedře počítačů, aby ti, kteří mohou, napsali svou diplomovou práci v jazyce anglickém. I mně to bylo doporučeno a já, poslušen rady dobře míněné, snažil jsem se zhostit daného úkolu, jak nejlépe jsem dovedl. Jistě to neznamená dokonale a čtenář anglického jazyka znalý nechť chyby laskavě promine. Naopak čtenář ne zcela zběhlý v četbě této novodobé latiny, bez které se dnes žádný vzdělanec oboru technického neobejde, nechť se nenechá jazykem cizím odradit a tuží mysl. Ani jeden ani druhý ať pak čtouce nezapomínají, že i česky se sluší v této zemi umět. Na naší škole mu to často připomínáno není.

A word to the English speaking reader

It has become a certain fashion at the Department of Computer Science and Engineering, for those being capable of it, to write their master's thesis in the English language. Following the advise of my supervisor, I have tried to compose one too. This impertinence of mine towards the language of Shakespeare and Poe shall be justified by the first person who reads this composition, but would not be able to do so if it was written in Czech. Also, I believe the native English speakers will excuse me - they must have got used to such a behaviour, as their mother tongue, having become a mean of international communication, cannot be considered as belonging to them uniquely any more.

Contents

I. Introduction

With the fast development of advanced examination techniques in medicine which we have been experiencing over the several past decades, there is an enormous need for tools to help the doctors understand the huge amount of data. Those of the doctors of medicine who do not work directly in the domain of the one particular examination technique need the data processed or evaluated by a specialist, e.g. a radiologist. As the methods are becoming more and more a routine, it is highly desirable that the data produced during the patient's check-up are processed automatically and presented to the medical expert in an easily treatable form.

This is particularly valid for *Magnetic Resonance Imaging* (MRI), a non-invasive examination technique, which has won a notable, irreplaceable position within the medical visualisation techniques since its introduction to practical use at the end of the 1970s. It was derived from magnetic spectroscopy, a much sooner known analytical chemistry method, based on the observation that different elements behave differently in magnetic field. F. Bloch and E. M. Purcell were awarded Nobel price for the experiments performed in 1946. But it was limited to laboratories until 1972, when Magnetic Resonance (MR) was first proposed as a tomographic visualisation method.

Like the earlier Computer Tomography (taking roentgen scans of the patient's body from multiple angles and computing the volume data), Magnetic Resonance provides three-dimensional information on the anatomical structures of the human body. There are two main advantages of MRI over the former method: it is believed to be harmless (no roentgen radiation is employed to acquire the data), and it can distinguish various kinds of anatomical tissues (unlike Computer Tomography, which is applicable almost uniquely to bones). All this, together with the boom of computers, has made Magnetic Resonance a standard diagnostic procedure.

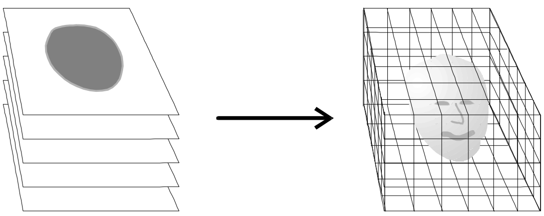


Figure I-1. This sketch symbolises the nature of MR data. The slices on the left can be understood as (or interpolated into) 3D voxel cube on the right. Attentive reader might have noticed the human head, hiding in the 3D data. Can we extract it from there?

The Czech reader is referred to the Žižka's article [48] for a good introduction to Magnetic Resonance from the radiological point of view, we will henceforward concentrate not on the ways and ingenious procedures of obtaining the MR data, but rather on their processing and understanding. We can start by discussing the properties of the data.

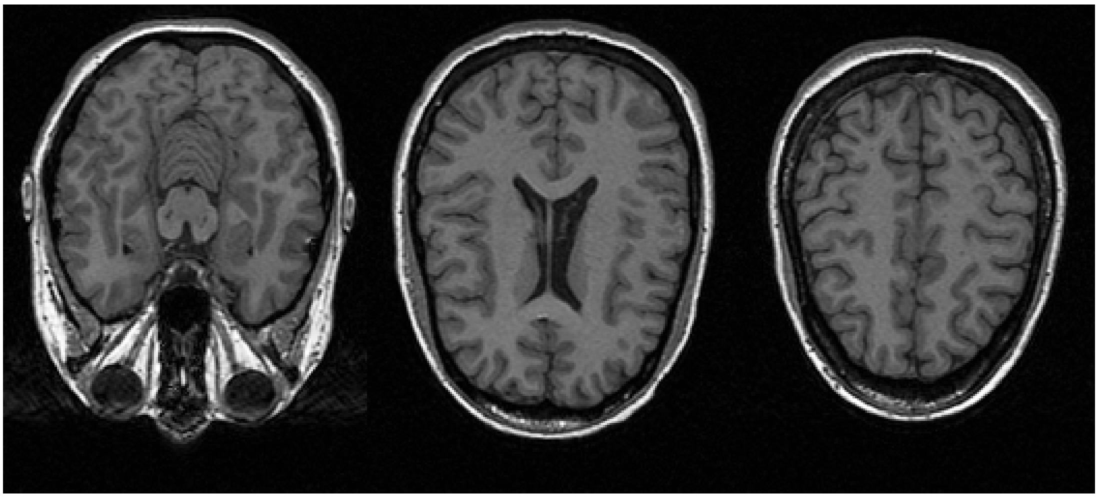


Figure I-2. Typical examples of MR images of the human head taken at (picture on the left) and above (middle and right picture) the eyes.

Magnetic Resonance yields the data as a set of 2D slices, which together form 3D cube of voxels - see Figure I-1 (some interpolation is necessary to obtain correct 3D data because of unequal element distances in directions of axes x, y and z). Typical dimensions of individual slices are 256x256 or 512x512, with modern MR scanners even 1024x1024 pixels, while the distance between neighbouring slices varies from 5 to 1 mm (all this was meant for MR scans of the head. Magnetic Resonance can be employed equally well for other parts of the body, but we have focused on brain data in our work). The data are still typically visualised as the set of 2D images, where the values of measurements taken from the patient are expressed through the brightness of grey at a particular position of the image. Examples of such 2D MR images of the head are seen in Figure I-2.

The tissue classes, such as white matter, grey matter, cerebro-spinal fluid (CSF) and fat are distinguishable according to their grey-levels. Some anatomical structures, e.g. brain ventricles (the X-shaped patch in the centre of the middle image) are well visible from the first sight, but other deep-brain organs, less contrast or smaller in size, are much more difficult to trace. Sometimes the 3D connectivity and information from neighbouring images could help, but it is hardly accessible to the medical expert examining the separate 2D slices. Full three-dimensional views should be useful to supply an idea on 3D nature and position of an anatomical organ with respect to others, a notion which is essential e.g. in neuro-surgery. But to be able to offer such a 3D view, we require that the anatomical structure to be visualised is extracted from the data, in other words we need to perform *segmentation* of the data first. To accomplish this, techniques from the domain of computer vision and related sciences must be used effectively.

Segmentation here means no less then separate the MR data into unique categories, where each category represents either a tissue class (*grey matter*, *white matter* etc.) or a distinct anatomical structure, e.g. *left ventricle* or *brain stem*. We can find either full segmentation, i.e. complete image understanding, or partial segmentation, extracting only few organs and omitting the rest. When aiming at delineation of structures rather than tissue classes, complete segmentation is practically impossible to achieve, considering the huge amount of fine and imperceptible anatomical details in human brain.

Understanding the 3D Magnetic Resonance data is a problem different from those of classical computer vision, where we wanted to find either two- or three-dimensional objects in 2D images only. In this sense, segmentation of MR images is more constrained and thus easier. On the other hand, being obliged to handle 3D data makes the task more complex. Unfortunately, the nature of MR data does not facilitate the assignment either: the images lack strong edges, classical methods like thresholding are useless as the acceptable grey-level values for different organs overlay, the values for the same anatomical structure may vary considerably according to the position in an image for a single patient (not speaking of the differences between individuals), the data are noisy. For all these reasons, any segmentation method relying on the input data only is bound to failure. It has proved highly advantageous, even inevitable to exploit some other source of knowledge together with the MR data, in particular anatomical knowledge on position and typical shape of the organs we are looking for.

Aware of the goal of the project, which is a method for automatic or semiautomatic segmentation and labelling of anatomical structures in human brain, we have tried to construct a framework allowing to represent anatomical knowledge. The method is based on shape model of an object's surface. But before we embark upon a detailed presentation of this model's virtues and weaknesses, we will review the approaches published previously to ours.

II. Preceding work

A huge number of articles on MRI data processing, segmentation or understanding has appeared in the last few years. The applications vary from simple, fitted onto one particular problem, up to general systems employing advanced methods of artificial intelligence, computer vision and geometrical modelling. However different the published approaches are, we will try to classify them according to the sort of information the application makes use of apart from the input MR data, and we will observe how that knowledge can be represented and implanted into the segmentation algorithm.

In the second part, we will notice whether the applications work with raw input MR data or perform some pre-processing usefully. We will also ask what is a method's relationship to space, i.e. whether it performs the segmentation procedure in the two dimensional slices separately or this limitation not hold and some three dimensional information contained in the input data is exploited as well. At the end of the chapter, few words on global or local methods and on additional refinement techniques will be found.

II. 1. Knowledge representation

As mentioned in the introduction, the properties of MR data make the traditional computer vision techniques of image segmentation, such as thresholding or edge detection, partially or completely unusable. Much more sophisticated methods are needed, methods which do not rely on the input data only, but exploit other information source, e.g. knowledge of anatomical structure of the objects to be found, information on its shape or common position. In this chapter we are going to classify the published approaches according to the way they store and employ such anatomical knowledge during the image segmentation process. However, the reader should be aware of the fact that such a classification does not claim to be doubtless, because the individual applications often use similar concepts at different stages of the process, so what is the basis of one, can form but a small complement in another segmentation procedure. But let's cease these introductory speeches, practical examples should clarify the obscure field of medical data segmentation better.

II. 1.1. Informal knowledge representation

This heading intended to express the notion of a “lack of formal system”. The applications do use anatomical knowledge, but it is inherently included in the segmentation process; they rely mostly on the colour of pixels, histogram or other low-level vision concepts. Some anatomical information is employed then to remove errors or to control the process through simple mechanisms, such as connectivity check.

The first segmentation system we heard about was **Pinta [32]** of Parvin and Johnston. It is able to segment brain into four categories: cerebrospinal fluid or bone (which are the darkest in the image), grey matter, white matter, and others (which could include tumours). First, the histogram[1] is analysed to estimate the average intensity of each region. This is followed by fuzzy K-means clustering (assigns probabilities that a pixel belongs to a class) and relaxation[2] of the patches and results in hypothesised regions within individual slices. The second step is then comparing the neighbouring slices to apply anatomical knowledge and ensure consistency: we know that white matter is a singly convoluted structure and that grey and white matter regions always share an edge. Any region violating these restrictions can be removed from the list of active hypotheses.

A method similar to that of Pinta was employed in [43], with the objection that this time the underlying scheme was Bayesian Decision Theory. But here as before, stochastic relaxation enforces local consistency. We refer the reader to the cited article for detailed explanation.

Another, simpler approach was presented by M. Bomans et al. in **[4]**. Their goal was to divide MR head data into few major organs such as brain, ventricles and skin. They combined the Marr-Hildreth operator for **edge detection** and **morphological filters**[3] for refinement of the detected edges. The morphological filter employed was *closing*, which is a combination of operators *dilation* and *erosion*. In general, it results in an object simpler than the original, with fewer details, smoothed contours and filled holes. In the end, the segmented objects were manually labelled and displayed. The whole process was, mind you, fully three-dimensional, but, on the other hand, there were two drawbacks: the necessity of manual labelling, and the fact that both the edge detector and the morphological filters are computationally very expensive.

A completely different method appeared in **[34]**. The author, S. P. Raya, created a **rule-based system**, which, through several low-level features, separates the voxels of brain into few anatomical categories. The rules were presented in algorithmic form like „if DENSITY_1 is HIGH and DENSITY_2 is MEDIUM then voxel is a LEVEL_2_MEMBER

---

[1] Explanation for a computer vision newcomer, histogram is „a vector H with the number of elements equal to the number of brightness levels. The value of each element expresses the count of pixels of the corresponding brightness in an image.“ [22] It supplies a global information on the distribution of grey levels in an image.

In the case of MR images of the brain, the histogram has ideally several peaks corresponding to the classes of anatomical tissues, such as white matter, grey matter, cerebrospinal fluid, bone, and high contrast objects (tumours).

[2] Relaxation, too, represents a technique frequently employed within the voxel-oriented applications. It is „an iterative scheme, where local properties are propagated and noise is removed by enforcing consistency“ [32].

[3] Mathematical morphology plays an important role in low-level computer vision. Czech reader is referred to the book [22].

of A_CLASS and GOTO RULE 250“. We must confess we have not tried to immerse in detailed study of this early technique.

We might mention another, more recent article **[24]**. A clone of the basic grey-level tissue classification mechanism, which should be able to compensate for inhomogeneities in the MR scanner, is introduced briefly. The classified regions are then refined by a deformable contour model, called snake. This advanced vision technique will be dealt with in later chapters.

II. 1.2. Abstract knowledge representation

Reminding the reader of the fuzzy character of the categories, we include here two techniques not belonging to the previous chapter because of the sophisticated mechanisms linking the segmentation process to some internal knowledge source.

Here we go: a **system for boundary detection in four-dimensional cardiac MRI sequences** is presented in **[38]**. The authors divided the process into three stages.

First stage, low-level processing, comprises rectangular-to-polar conversion (as they are concerned with heart segmentation, where most objects are round in shape, this conversion usefully substitutes lines for circles; lines are obviously much easier to find), grey-level histogram computation for each slice, and initial estimate of the region boundaries. Second stage applies grey-level and location knowledge source to assign each pixel a set of plausibility function values, which describes the probabilities that a pixel belongs to a particular class (three classes appeared in the application: blood-pool, muscle and lung). The third stage, high-level processing, employs Dempster-Shafer theory of evidence as a framework for combining information that is uncertain, imprecise and inaccurate to link the image and some a priori knowledge and thus accomplish labelling of the image objects. The information obtained is used to assist labelling in temporally and spatially adjacent slices.

The second approach in this section, the **graph model of Robinson, Colchester and Griffin [35]**, is going to be described in a larger scope to give the reader a fair idea of its sophisticated symbolic model representation.

The model is stored as a frame based semantic network consisting of three coexisting graphs (a spatial adjacency graph, a part hierarchy and an inheritance graph). It was implemented to segment 2-D MR images of the brain.

Every individual anatomical feature is assigned a frame consisting of a series of slots. The slots and their values provide an object-centred description of the structure in question. The most important slots are the anatomical name of the structure and the „shape“ slot, based on the Delaunay triangulation and the Voronoi diagram. Each frame has a slot describing the position of the centre of mass of the object, and a slot stating the tissue type of the object it describes. This information can be used to predict an approximate grey-level or range of grey-levels that the object is likely to have in the input image. Relations between structures are coded as links between individual frames, and these links are stored in form of a pointer to another frame in a special „relation-slot“.

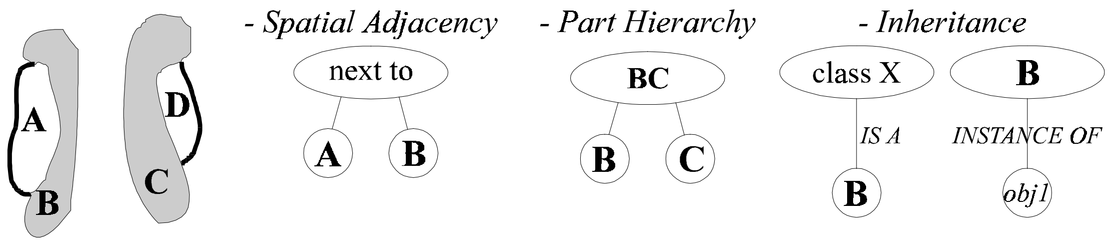


Figure II-1: Examples of relationships represented in the graph model.

The links connecting individual frames represent a semantic coding of the anatomical knowledge. There are three types of links between structures:

- links providing a series of spatial adjacencies between structures
- links providing a hierarchy of parts/sub-parts
- links providing a network of inheritance properties

These links form three coexisting graphs within a single network (see also Figure II-1).

Spatial Adjacency Graph is used because relative positions (expressed in terms such as "posterior", "superior" or "left-of") of objects within the brain may differ when abnormalities are present. What tends to remain constant are the relative adjacencies of structures. Based on spatial adjacencies, hypothesis about particular objects extracted from the image can be made and then verified using more complex criteria.

Part Hierarchy Graph specifies anatomical features in terms of their parts and sub-parts. Many anatomical features form complex three-dimensional shapes within the brain. These complex shapes are often difficult to extract as a single entity. However, they can be divided into smaller constituent parts which are easier to recognise. Each part/sub-part of a structure is represented by a single frame in the same way as for complete anatomical structure. The use of part hierarchy makes the expressing of the relationships between structures much easier. Spatial adjacencies need only be expressed between features at the same level in the part hierarchy; spatial relationships between features at different levels can be inferred as required. Certain other types of information, such as tissue type, can in some cases be inferred as well.

Inheritance Network. The final coexisting graph within the model. This network allows frames to inherit values for slots if there are no values attached directly to the frame in question. There are two types of relations and a clear distinction must be made between them not to confuse the long-term knowledge coded in the model and the image dependent information.

The "IS-A" relation is used to connect *model* entities to the more general class of a model entity, e.g. *"left-frontal-lobe IS-A lobe"*. Information general over the whole class is stored in the more general frame and inherited by the more specific occurrence of the class *within the model.*

The second relation used to define the inheritance network is the "INSTANCE-OF" relation. It provides labels for features *extracted from the input images.* For example *"obj1 INSTANCE-OF left-frontal-lobe"* provides a label, i.e. a link to a specific model entity for an object extracted from the input data, and consequently joins the object to the knowledge coded within the model.

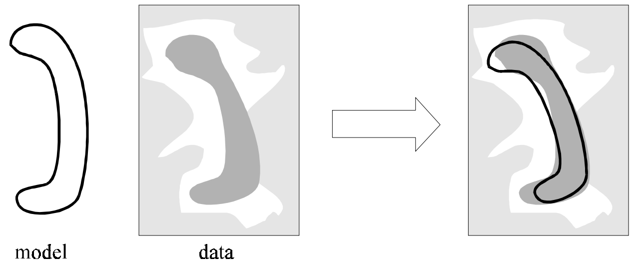


Figure II-2. Image segmentation through maximisation of the fit model to data. How much deformation do we allow?

When employing the symbolic model for identification of objects in an image, edge detector is applied first. From the edges found, objects are formed and description created for them. Then, these image objects are compared with the objects in the model and if a good match is encountered, hypothesis labelling the image with the name of the model object is founded. At the last step, reasoning mechanism merges and otherwise develops the hypothesis to obtain few consistent final solutions. More detailed description of the reasoning process and some comments can be found in [30]. Some aspects of the method will be discussed later in the section on data pre-processing.

II. 1.3. Surface Models

Innumerable applications in the field of Magnetic Resonance data segmentation are based on shape models. The idea behind this is quite simple and copies to certain extent human behaviour. Wanting to localise an instance of a known object in an image, one would try to look in the subpart of the image, where the object is found „usually“, and search for colour, shape or boundary typical for that object. Similarly, only omitting the inside of an object and relying on the boundary alone, the shape model approach attempts to identify such a position in the MR dataset, where the image best corresponds to the stored model appearance. This is often performed through placing the parametrically deformable model into the image and finding such a set of the model parameters, for which the match between the data and the model is maximised. Another possibility is to have a rough approximation of the object boundary and try to delineate it more accurately by allowing the boundary to move around and deform under some physically defined restrictions. Although the two methods are frequently usefully combined, we will examine them here separately first.

Geometric Shape Models

As sketched in the introduction above, using parameterized geometric shape models we transform the problem of finding an object in the image into a qualitatively different task of maximising the fit between the model and the image data through varying the model parameters. The advantage of this approach lies in the fact that the model, being controlled by the parameters, is not allowed to deform into an unfeasible shape, and therefore can be robustly applied to segmentation of incomplete or noisy data. The model can catch only part of the boundary and make out the missing parts. But equally important, an anatomical object can acquire a large variety of different complex shapes, which are difficult or altogether impossible to describe by a limited number of model parameters, and the model can hardly yield perfect solution, especially when abnormalities, not encountered before, are present in the input data. Recognising that this problem of limited knowledge is common to all

„artificially intelligent“ learning systems, we can start with the discussion on how the shape model can be represented.

Several authors (see [44], [16], [41]) have based their model on the geometry of **superquadrics**. A superquadric is a geometrical primitive, which can describe many different 3D shapes: the family of superquadrics includes superellipsoids, superhyperboloids and supertoroids. The practical applications relied on a conventional superellipsoid. It is described by following parametrical equations:

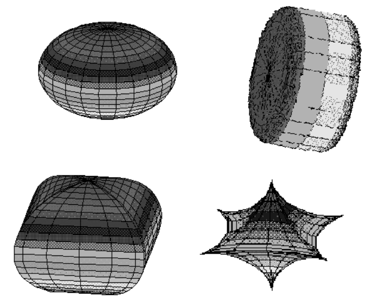


Figure II-3. Examples of superquadrics.

$$\mathbf{s} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = a \begin{pmatrix} a_1 C_u^{e_1} C_v^{e_2} \\ a_2 C_u^{e_1} S_v^{e_2} \\ a_3 S_u^{e_1} \end{pmatrix}$$

where $-\pi/2 \le u \le \pi/2$ and $-\pi/2 \le v \le \pi/2$,

and where $S_w^e = \mathrm{sgn}(\sin w)\left|\sin w\right|^e$

and $C_w^e = \mathrm{sgn}(\cos w)\left|\cos w\right|^e$.

Varying the set of parameters ($a, a_1, a_2, a_3, e_1, e_2$) in the equation above, a wide range of 3D shapes can be produced (see Figure II-3). These *global parameters* can serve for capturing gross shape features from visual data or for indexing into a database of objects if classification is concerned. However, these few parameters produce only regular geometrical shape and cannot reconstruct details of the usually irregular and complex shape of anatomical structures. Therefore some additional techniques must be employed to improve the elasticity of the model.

In [41] Terzopoulos et al. complemented the global parameters of a superquadric with local parameters of membrane splines to create a new family of hybrid models, dubbed *deformable superquadrics*. Position of points on the surface of the deformable superquadric can be expressed as the sum of a reference shape $\boldsymbol{s}$ (classical superquadric) and a displacement function $\boldsymbol{d}$:

$$\boldsymbol{p} = \boldsymbol{s} + \boldsymbol{d}$$

Here, the displacement function supplies the local parameters to the model and serves for smaller deformations, irregularities and captures details of the modelled object. Figure II-4 illustrates the model geometry.

The same idea only with different mathematics appeared in [44]. In this case, the local deformations, or claimed smoothly transformable global to local deformations, were constructed in orthonormal wavelet bases.

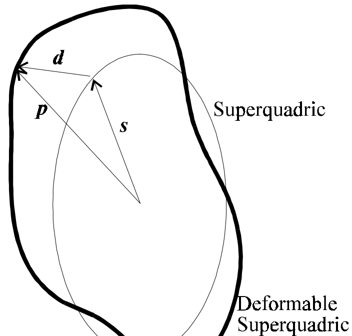


Figure II-4. Geometry of the Deformable Superquadric

Another combination of complementary approaches was presented in [16]. The superquadrics there provides only an approximation, an envelope within which the object will be found, and then histogram-based procedure extracts the object itself.

When deforming the model to fit the visual data, the superquadric is subject to a set of physical equations: kinematics, dynamics and elasticity of the model are expressed. The input image data are transformed into forces acting on this physical model. So, for example, edges in the image may attract the model boundary. Iteratively, through varying the parameters, we strive to minimise some globally defined energy of the solution.

To discuss briefly the presented results: unfortunately, the method presented in [41] was not used for segmentation of medical images. The authors of [44] employed their model to extract gyrus and hippocampus from the MR brain data, and although they claimed their model to grant the power of continuous local to global deformations, the published results seem coarse and not very detailed. The hybrid scheme of [16] might represent a feasible solution of globally found approximation and detailed tissue separation inside the superquadrics boundary. However, limitations of the histogram-based methods hold here: the object has to be distinct in colour from the surroundings.

Well, to conclude the superquadrics section: they seem very elegant in that they need only a small number of parameters to describe a shape „completely". A problem inherent in the use of a superquadric-based model lies in the rather complicated mathematics and resulting large number of floating-point operations.

A distinct category must be devoted to **Point Distribution Models** - PDM ([8], [20] and many others). Various members of Department of Medical Biophysics, University of Manchester, including T. F. Cootes, A. Hill and C. J. Taylor, have over the past few years developed first 2-D and then also 3-D statistically trained models of the shape and appearance of flexible objects (such as organs) seen in images. As we based our approach to image segmentation on their model, most aspects are going to be discussed in detail in the chapters to come. Only few basic ideas are mentioned here.

The object shape is represented by a set of connected points forming the model boundary. The model is trained on N examples of an object, and through analysis several main modes of deformation are found and stored. The grey-level appearance is stored for a small vicinity of each point on the boundary (see Figure II-5). These grey-level profiles have one big advantage: the model boundary is not necessarily formed by strong edges and subtle changes in colour can be recognised and located in the image.

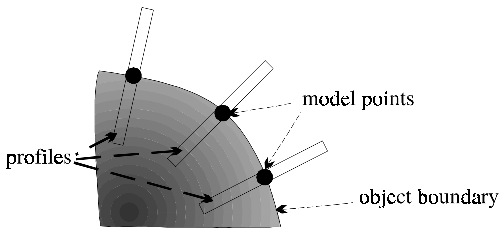


Figure II-5. Part of the Point Distribution Model: connected points form the model boundary, grey level profiles represent the object appearance.

When looking for an instance of the modelled object in an image, two distinct methods are applied. First, *Genetic Algorithms*, global method of optimisation, are employed to provide approximate location of the model template (again, see detailed description later on). Then a second, local technique of tuning the model parameters, dubbed *Active Shape Models*, is used to refine the hypotheses. Every point finds the position in its surroundings, where his appearance best corresponds to the image. Then a new set of the model parameters is computed and the model points' positions updated, so that the new points are as close as possible to the suggested points.

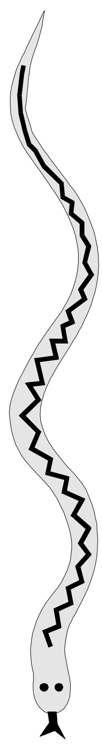
In [8] the Point Distribution Models were applied to segmentation of echocardiograms, locating the prostate in MR slice of the lower abdomen and tracking brain ventricles. All these applications were in two dimensions. The extension to 3D was introduced in [20], where a more complex model of skin, ventricles, brain stem and caudate nucleus was created from thirty segmented training examples. Once more, we defer more involved discussion till Chapter III.

The last method of boundary model representation to be mentioned in this section: published in [46], this 2D model was based on Fourier decomposition of the closed boundary shape. The shown experiments proved successful in depicting corpus callosum in MRI of brain and some parts of heart, only the model was initialised very close to the final position.

Deformable Contours - Snakes

Imagine the following situation: you have a contour of an object which you would like to trace in an image quite near the desired position (it was placed there either manually or by a method from the previous sections), and you want the boundary to be shifted and deformed (reliably, without self-intersections and other malfunctions) to fit the actual object border as accurately as possible. Well, if this is the problem you are facing, snakes and their extensions might offer a solution.

With the name given by Kass, Witkin and Terzopoulos in [25], probably the most quoted article we have noticed, the method of „snakes" has gained a large number of followers within the computer vision community. The principle is straightforward: the models imitate the real behaviour of snakes in nature and through crawling, creeping, bending and turning achieve the optimal position in the image. Unfortunately, the physical and mathematical apparatus behind that is more intricate.



Originated in 2D [25], a snake was a planar curve, which had the initial position (specified interactively by the user) and an *objective function* associated with it. This objective function, referred to as the energy of the snake, is expressed:

$$\mathbf{E}_{snake} = \mathbf{E}_{internal} + \mathbf{E}_{external}\,.$$

The internal energy of the snake can be interpreted as energy (or cost) of deformation, serves as regularisation force (constants of elasticity and rigidity take place in its computation), and depends on first and second order derivatives of the curve, whereas the external energy attracts the snake to interesting features in the image, typically to locations of maximum gradient. The best position of the snake is then achieved through finding local minima of the energy $\mathbf{E}_{snake}$.

The **extension of snakes into three dimensions** was published by Cohen and Cohen in **[6]**. They formed self explaining *balloon model*, whose governing equation comprised one member additional to the classical equation of snakes: inflation force pushed the contour from inside thus preventing it from collapsing into a single point. The model was claimed to outline an object correctly, having started from almost any shape inside the structure of interest. The article suggests finite element method for solution of the snakes equations.

Note that the snake or balloon movement in the articles summarised above is free except for the internal energy factor. However, it might be reasonable to constrain the deformations globally. Such a solution is found e.g. in [8], mentioned in the previous section. During the refinement stage of *Active Shape Models* (dubbed also by the authors „smart snakes"), the model is allowed to deform only in ways learned from the training set. This makes the search more robust and resistant to misleading forces.

A similar idea was applied in **[39]**. But, unlike the Point Distribution Model of [8], where the shape was parameterized by displacement of corresponding points on the surface, their model was built on **Fourier Parametrization**. In the 2D example presented, they took 30 segmented examples of corpus callosum, expressed the closed boundary of each of them in terms of Fourier series, and performed *principal component analysis*[4] of the covariance matrix of the first 100 Fourier coefficients, thus obtaining 12 eigenmodes, i.e. 12 main ways of deformations found in the training examples. The model built this way was placed rigidly into the region of interest using Hough transform, and then restricted deformation was allowed to refine the solution.

The model was extended to 3D, where the object surface was described by spherical harmonics functions. The principles - model building, rigid placement and controlled elastic deformation through minimisation of energy - were exactly the same as in the 2D case, the main problem being a uniform mapping between the object surface and two parameter space for model representation. The technique was successfully applied to segmentation of deep grey matter structures of brain. Again, the model was trained on 30 examples.

Similar application of Fourier decomposition to model representation was published in [46], but this time limited to 2D only.

[4] Principal component analysis is a frequently used method (e.g. [8], [39]) for analysis and extraction of most important variations or differences present in a set of examples of an object. Having the covariance matrix of the examples, the main variation modes can be found as eigenvectors corresponding to the largest eigenvalues of the matrix. Involved explanation of the mathematical background is presented in book [50]. If the reader isn't familiar with the terms eigenvector and eigenvalue, he is referred to any textbook on linear algebra.

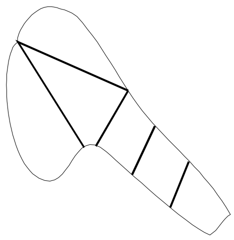


Figure II-6. In [31], the links between non-neighbouring nodes introduce volumetric properties into the model.

A different suggestion how to represent object deformations was made by Nastar and Ayache in **[31]**. To track and analyse non-rigid motion of dynamic structures in time sequences of 2D and 3D medical images, the authors designed a **physically based deformable model**. Object was considered as a mesh of masses linked by springs. *Equilibrium forces* act on the object to conserve its initial shape, while *external forces* attract the boundary points to landmarks or regions of interest in the images. The authors propose also not to restrict the model points connections to the object surface, but add some cross-object links as well, for these would help the model maintain the correct shape and simulate its volumetric elastic properties (see Figure II-6). The solution presented leads to a set of linear dynamic equations, decoupled for each co-ordinate, of reduced algorithmical complexity compared to classical snakes.

Some of the approaches presented above tend to reduce the freedom of deformations of a snake thus avoiding unfeasible solutions. McInerney and Terzopoulos went in **[28]** the completely opposite way: they constructed a framework allowing their snakes to change topology, flow into complex shapes, split and merge. Their **topologically adaptable snakes** are therefore particularly suitable for segmentation of complex shapes, such as arterial trees. On the other hand, being freely deformable, the snakes require good contrast between the object of interest and background.

Another very interesting application of snake-like approach appeared recently in **[36]**. An **atlas** in form **of labelled points on the surface of a normal brain** was presented there. The surface was expressed in terms of 3D basis functions, B-splines. The model representation resulted in a mesh of about 5000 points. To apply the atlas to brain surface delineation, pre-processing was performed first. This consisted of boundary finding and mathematical morphology, and resulted in smoothed brain extracted from skull and the background. At the next stage, an extension of snakes to 3D deformable spline models took place and the model was warped onto the actual brain data. The authors claim the approach to be robust (which is supported by the use of pre-processed, separated and smoothed surface - snakes are less likely to converge to false local minima of the energy).

II. 1.4. Volumetric atlases

In contradistinction to the previous sections, where the objects were represented only by their boundaries or surfaces, the attempts to model anatomical knowledge in its truly volumetric nature will be reviewed here. We might mention several transitional surface-to-volume approaches at the beginning: one of the last methods mentioned in the snakes section employed volumetric links crossing the object volume and connecting distant boundary points; extensions to Point Distribution Models ([2], [27]) usefully included information on grey-level appearance inside the object into the model. We might also note that some of the applications based on histograms, mentioned in the first part of this chapter on knowledge representation, processed real volumetric anatomical information too. However, the following systems of volumetric atlases are qualitatively different.

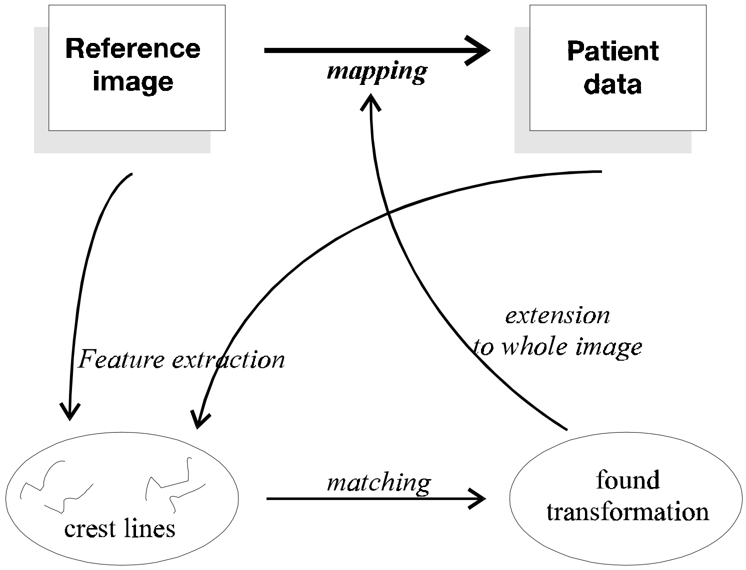


Figure II-7. Segmentation through a mapping of the labelled reference image onto the input data, as implemented in [14].

Before we present the atlases for data segmentation, we may make a little digression and mention one project from the domain of human anatomy visualisation. A group of German researchers around Schubert and Hoehne (see [37]) designed system **VOXEL-MAN** for computerised presentation of anatomical and functional structures of human skull and brain. They manually, from expert segmentation of MR data, through volume editing and other knowledge acquisition, constructed representation that links visual and symbolic knowledge into one framework, so called intelligent volume. A student of human anatomy is by means of their program allowed not only to visualise parts of the brain in three dimensions, but also to explore more information sources, e.g. description of the object, its position within the symbolic tree of anatomical structures or within the magnetic resonance images. The researcher can select parts to be visualised manually or logically, through selection of all elements taking part in some physiological function[5]. But despite these virtues, the architecture, having been planned for presentation purposes, is not adapted for automatic segmentation of unseen MR datasets.

Segmentation using volumetric atlases faces a problem similar to that of *registration*: if we take more than one set of medical data for one patient, e.g. data from MR scanner T1 and T2 weighted, or MR and CT images, the data will most probably occupy different position in space due to patient's movement or inconsistency in scanners. It is then necessary (to obtain multiple corresponding measurements at every point) to match individual sets of data through translation, scaling, rotation and slight non-linear deformations. Back to atlas-based segmentation, the task is obviously more difficult because we are concerned with *inter-patient registration*, where much more deformations and differences can be encountered than in the case of multiple data of a single patient.

An example of such a registration is found in **[14]** and **[15]** by Declerk et al. Segmentation of a patient's dataset is accomplished by automatically generating the **mapping between a completely labelled reference image and 3D medical images of the patient**. To achieve this, three techniques are combined: first comes extraction of 3D feature lines from both the

[5] There is one installation of VOXEL-MAN at the Computer Graphics Laboratory, so the reader having access there is encouraged to have a look.

reference and the patient's data. As features, crest lines were employed, i.e. lines formed by successive points of maximum curvature on isosurfaces of organs (isosurface is surface of equal grey-levels). Second step consists of matching the crest lines using 3D deformable line models (this is done via point matching and least square detection of second order polynomial transformation which maps the atlas crest lines onto crest lines of the patient's data). The last step is then formed by extension of the found deformation to whole image space using B-spline warping techniques. Having mapped the reference image onto the processed image, the atlas labels are mapped and complete segmentation is accomplished (see Figure II-7 for a sketch of the procedure). We could mention that the authors propose 3D surface snakes to refine the solution if more precise final segmentation is needed. Advantages of employing crest lines prior to surface models lie in more compact representation, easier manipulation and less ambiguous point correspondence.

Another ingenious segmentation system built upon volumetric atlas was introduced by Collins et al. in **[7]**. Similarly to the application described above, the approach depends on 3D digital model of human brain anatomy that contains both volumetric intensity-based medical data and geometric atlas data (labelling) that coexist in a brain-based co-ordinate system - *stereotaxic space*[6]. And as well there, segmentation of an image is achieved by mapping of these reference data onto the dataset to be understood. The mapping (or, again, *registration)* procedure is iterative, hierarchical and non-linear in terms of permitted deformations. In contrast from the previous method, this one does not rely on crest lines extraction, but accomplishes the registration using other invariant features of image intensity and gradient magnitude. The optimisation to recover the best transformation is performed in hierarchical multi-scale fashion, which increases speed and robustness.

It is apparent that the reference atlas, which is employed for mapping, should be as similar as possible to all instances of the input images for best results. The MRI atlas in the previous paragraph was built to represent the most average from 305 (!) normal brains in stereotaxic space. The authors reported in the article [7] that they were constructing a probabilistic atlas of neuro-anatomical structures, where each voxel would represent the probability of finding a given structure at that voxel position.

The final method to appear in this section was published by Mangin et al. in **[26]**. It suggests to base the low-level processing of 3D medical images on **deformable topological models**. Once more, the model is an image consisting of labelled regions, and segmentation is completed through topology preserving deformations of the model onto the data. The deformations aim to minimise some global energy and obey the following constraints: homotopy preservation (region connected components, cavities, tunnels), connectivity preservation and adjacency interdiction. The approach is particularly suitable for dividing data into the tissue categories like grey and white matter, ventricles, CSF, skull, skin, void.

[6] A neat definition is found in [7]: „*Stereotaxic* refers to the representation of 3D brain structures in a standardized brain-based coordinate system [40] such that all brains have the same orientation, position and size in the three orthogonal directions. Mapping volumetric data to this coordinate system permits comparison of brain from different individuals as well as to the published Tailarach atlas."

II. 1.5. Knowledge representation concluded

The reader must be now anxious to ask the obvious question: which of these many representations should I choose if I intend to employ anatomical knowledge in my application? Unfortunately, we can offer only one doubtless though unsatisfactory answer - *it depends*... It depends on the task to be accomplished. If separation of tissue classes is the case, you might employ some of the histogram-based methods with topological constraints. If you intend to manage complete classification of all structures in brain, you might find a volumetric atlas the only admissible solution. If you do not care so much about completeness but want to extract one organ of relatively stable shape but indistinct appearance, whole atlas would be an overkill (very expensive both for preparation - manual segmentation for training of the model is a tedious task - and computationally), but surface models with advanced representation of boundary appearance may prove useful. If, on the other hand, you would like to trace abnormalities, you might be forced to forget about complicated models at all, for all the models and atlases we have encountered yet were constructed for normal brains. As well as in computer vision, complete understanding of *any* image is an excessive task at today’s level of science and technology. Models can be employed helpfully where there is a small variation between instances of the same object. They are almost sure to fail for an instance that differs largely from shapes they were trained for, and this behaviour must be considered correct - the pathological object does not fulfil the required constraints. However, even this failure should be useful as well, warning us of the possibility of a problem, which can be afterwards judged by an expert.

To summarise the chapter in one sentence: the use of anatomical knowledge for segmentation of magnetic resonance data is inevitable, the particular task must be kept in mind when taking the decision of how to represent the knowledge.

II. 2. Several notes on employed techniques

We will, in the following short chapter at the and of the review of earlier work, attempt to discuss several interesting points which otherwise could have been passed unattended, distributed in various sections of the previous part. We will make few remarks on data pre-processing, 2- or 3-dimensionality of the applications, and on refinement of the segmented structures.

II. 2.1. Data pre-processing - gain or loss ?

The reader might have noticed in the chapter on knowledge representation that some of the applications work on raw magnetic resonance data, while the others perform some kind of pre-processing and then work only with sparse features extracted from the image data. Which of the two should be preferred?

Figure II-8 shows the difference between the two approaches. On the left, segmentation is accomplished directly on MR data. The trouble with this approach is the large amount of data to be worked with. Pre-processing, shown on the right, can decrease this amount dramatically, but an important drawback is inherent in this process: the feature extraction becomes the crucial part of the process. If we make a mistake at this first stage, and this is often the case when employing such features as edges of noisy MR data, there is no way back, the following segmentation step will be provided with incorrect input data and that can hardly be overcome by a posterior refinement. The problem is that the pre-processing is usually a local operation taking only a small vicinity of a point into account, so an important part of the information contained in the image could be lost. We must be aware of the sad truth that, according to Shannon's theory of information, processing of the data cannot create new information measurable by entropy. It can only accentuate or suppress some of the aspects and so reveal relations hidden in the images.

But here again, we must say that the decision will be specific to the particular task. For example the applications mapping volumetric atlases onto data are forced to perform feature extraction, because it would be otherwise computationally too expensive to obtain the desired deformations from complete volume data. Sometimes the global and local techniques can be combined usefully, and models based on object surfaces serve as an example of such a compromise: although they represent the shape globally, only a relatively small area about the boundary is taken into consideration when positioning the model in an image.

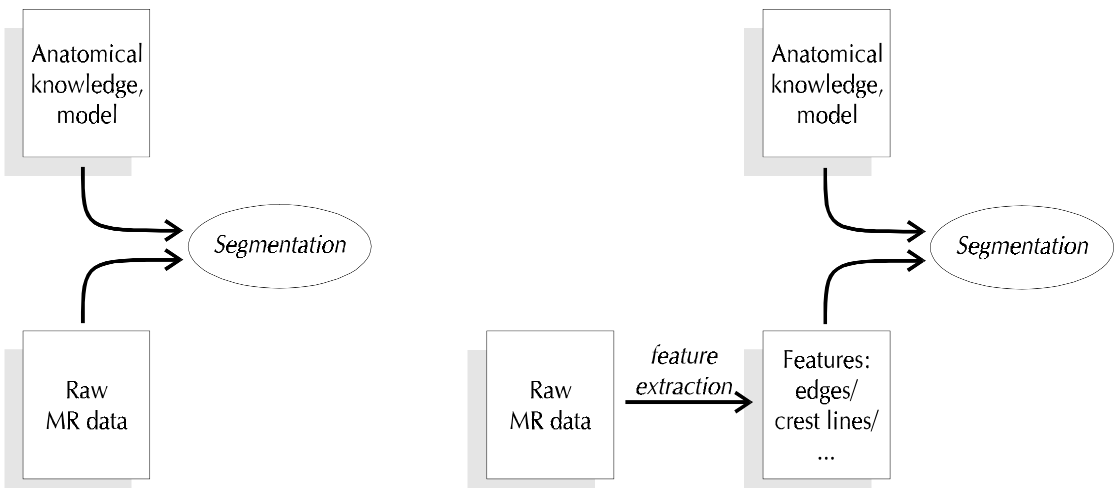


Figure II-8. The left schema figuratively shows the segmentation procedure performed on raw MR data, while the right sketches segmentation preceded by feature extraction.

II. 2.2. How many dimensions ?

In contradistinction to the discussion on pre-processing, judgement is straightforward at this point: full three-dimensional methods should be more powerful and stable then the two- or 2.5-dimensional, because through 3D connectivity some information can be propagated from the neighbouring slices into the one, where the image self does not provide enough evidence for reliable segmentation. The development in the medical data segmentation domain proves this assumption. The early applications processed each slice separately (see e.g. [35], boundary representation in [46] or snakes in [25] or [28]), later works still relied mostly on 2D slices, but employed 3D character of the data to check correctness ([32]) or to accelerate the segmentation by providing a good estimate in the adjacent slice ([8]). The latest articles reveal uniquely three-dimensional methods: voxel-based [24], snakes renamed to balloons in [6], other surface models in [31], [20], [39] or [16], and volumetric atlases of [14] and [7].

However, we don't want to condemn the 2D methods completely. It is clear that the mathematical computations (esp. geometrical transformations) are more complex in 3D, and if a two-dimensional method can manage the task, why not to employ it?

II. 2.3. Global optimisation and local refinement

Two contradictory forces play their role when taking decision between a global and a local segmentation technique: *global model-based methods* which employ anatomical knowledge reliably are generally more robust and exhibit a good resistance to false peaks of an objective function or to spurious edges in the image, but acquire only a coarse solution. On the other hand, the local techniques are capable of delineating the object with only small errors, but are not excessively stable and require a good initialisation to be successful. There is no need to emphasise the usefulness of a hybrid approach, combining the global and local solutions into a single system, where the global part provides a reasonable starting point for the refinement by a local technique.

Several examples of such a combination are found among the applications reviewed in the chapter above. So [4] finds the approximate locations of the objects boundaries by an edge detector and then shifts them and removes errors by morphological filters. The authors of [24] combine even more approaches: they start with a histogram-based segmentation, morphological operators follow to ensure local consistency, and 3D balloons-snakes finish the task as model-based correction (here the last step is meant as a robustness seeking complement rather then refinement). Snakes performed the opposite, refinement task in [14], where more robust and rough volumetric atlas model accomplished the initialisation step. The last to be mentioned, [16] allowed superquadrics to provide only an envelope for the second, histogram-based stage.

III. 2-D shape model

This chapter embodies introduction to our approach to the magnetic resonance data segmentation. Properties of the two dimensional model we have implemented are discussed and the accompanying techniques, allowing to employ it for image segmentation, reviewed. Results on both artificial and real MR data are displayed. Having explained the principles of our method in 2D, we will reveal the extension to three dimensions in the following chapter.

The model we have employed for MR data segmentation is founded upon the ideas of *Point Distribution Model* (PDM), introduced by a group of researchers from Manchester and reviewed briefly in the chapter on geometric surface models. Many references to their approach will be found within the text, and our methods will be thoroughly compared to theirs. We will try to emphasise the points where the two systems differ and where they coincide. (To distinguish the comparison-oriented paragraphs, we will print them *in a different typeset*).

III. 1. Model representation

The Point Distribution Model [8] is constructed from a set of N examples of a particular object, manually segmented from MR images. Every object instance in the training set is described by a labelled set of points, so we can write the n points of the i^{th} object in form of a vector $\mathbf{x}_i$:

$$\mathbf{x}_i = (x_{i0}, y_{i0}; x_{i1}, y_{i1}; \ldots x_{in-1}, y_{in-1})^T,$$

where (x_{ij}, y_{ij}) is the j^{th} point of the i^{th} shape. Each labelled point represents a particular position on the object, corresponding points on different objects represent equivalent locations on each object[7]. Then, the sets of points can be aligned and analysis performed. This includes finding the *mean shape*, calculated as the average of individual shapes:

---

[7] It is clear that the training objects need manual extraction from the data - this way the knowledge of an expert is brought into the system. At the beginning, the boundaries had to be labelled manually as well, i.e. an expert specified *corresponding* points, *landmarks*, on different objects' boundaries. In [21] a solution was proposed to overcome this tedious task: the 2D model is generated automatically from a training set of object boundaries. The algorithm finding the shape correspondences is shown to deliver better and more compact models than were those manually depicted.

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=0}^{N} \mathbf{x}_i$$

and finding the most significant *modes of variation* (as well as with another application reviewed in the previous chapter, it is accomplished using *principal component analysis*. See [8] or the footnote on page 13 for more details). Each mode of variation is described as a vector of displacements of individual points. Any shape in the training set can be approximated using the mean shape plus a weighted sum (weights = parameters) of these displacement vectors. Consequently, the model is allowed to deform only in ways found in the training set and no unfeasible shapes are generated during the image search.

We have to stop here and make the first comparison. We construct our model differently in that we do not obtain the admissible deformations from the training set and we do not include them into the model representation directly. The main advantage of such a choice is that we do not require large amount of training data. We will make experiments later on to see how much of the deformations found in medical data can be captured by a rigid model, and we will discuss additional techniques that are suitable for modelling plasticity. For the moment we will stick to the statement that we have only the mean shape, represented by connected points on the object boundary, without any deformations.

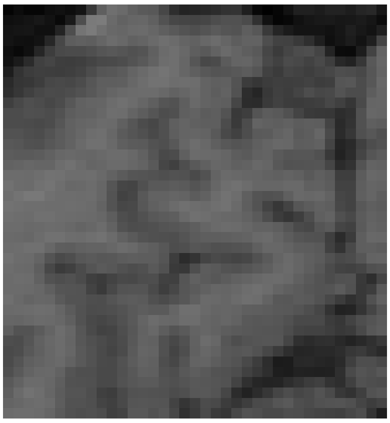


Figure III-1. A section from MR image on figure I-2 left, magnified to 400%.

When intending to employ the model for data segmentation, we cannot do with the object's shape alone, we require some information on the **object's appearance** too. It could be in the form of an interval of acceptable grey levels for a voxel-oriented technique. Most of the surface models restrict the model boundary to lie on edges in the image.

Figure III-1 shows a small part cut from a MR image and magnified. Do not blame low quality printing if you cannot see anything. The picture wasn't blurred artificially either, this is really how MR data look like - the boundaries between organs are rather fuzzy and no clear edges are found between them. In this situation, seeing the limitations of edge reliability, a very important novelty was implanted into PDM. This breakthrough, which has been proved to yield results superior to those of an edge-based search, is the use of *grey-level profiles*.

The idea is quite simple: an edge does not provide enough information and stability, we require a more general model of appearance. As we employ connected points to represent the boundary, we would like the stored appearance to be somehow linked to these points. Resulting solution could be to sample an area around the model points and store it as the model appearance. The authors of PDM decided to sample the image along short lines, each of them being centred at a model point and directed along the normal to object boundary at that point. Here again as everywhere else in human life, we are forced to a compromise by two contradictory forces: the longer the scanned line (dubbed grey-level profile), the larger parts of the original object's neighbourhood it stores and the better stability it shows, but the more expensive the computations are when comparing the stored appearance to a particular MR image. A detailed analysis of how the reliability and accuracy of the model-to-image fit depends on the profile length, but also on the profile type has been published in [9]. Not to neglect an important feature, we have to mention that the grey-level profiles are trained from instances of the training set and, as well as in the case of model shape, the mean and the typical variations can be computed and stored for them.

Again, to compare the Manchester appearance representation with ours: after short experiments with edge-based search, we embodied grey-level profiles in our model. Following the conclusions and advises of [9], we have constructed most of our models by extracting profiles of length 7 pixels from the reference image and storing them for the model points in form of normalised derivatives*. It was shown in [9] that this type of profiles yields the best results. The normalisation makes the profiles insensitive to addition of a constant (brightness) while the derivation ensures resistance to scaling of grey-levels (changes of contrast).*

III. 2. Image search

The principle of applying the PDM to finding an instance of the modelled object in the input image remains the same as for any other example of segmentation based on a shape model: the model is placed in the image in the location and with such parameters that maximise the fit between the data and the appearance stored for the model. To determine this location and parameters, two stages, dissimilar in approach and goal, take place. The first applied, Genetic algorithms, search the problem space globally to yield an approximate solution, and then a local technique similar to snakes refines it locally. Both these techniques play an important role in our project - let us examine them in detail.

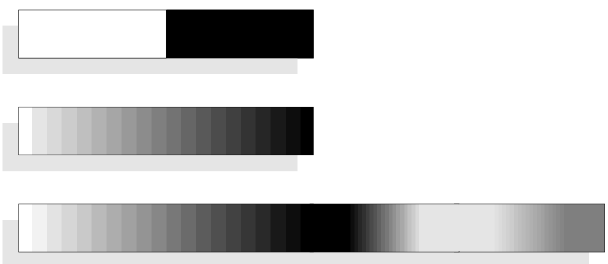


Figure III-2. Compared to the simulated edge at the top, grey-level profiles have the capacity to express the object's appearance more faithfully.

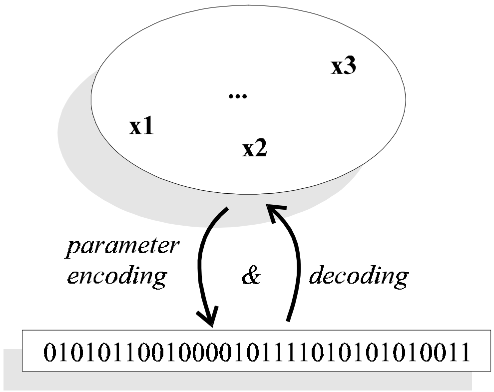


Figure III-3. Parameters of the objective function encoded as a bit string, chromosome.

III. 2.1. Genetic Algorithms

We have mentioned just above that performing the model-based image search could be understood as maximising the fit between the model and data. Now if the model is parametrical (i.e. its position in the image, its scale, rotation, shape etc. can be described by a set of parameters) and the model-to-data fit can be expressed by a *fitness function* of these parameters, the problem of image segmentation is converted into the task of optimising the function. This fitness, or *objective function* is likely to be noisy and discontinuous, of many parameters, and therefore unsuitable for classical, calculus-based optimisation techniques. We need a method capable of treating complex and large parameter space, insensitive to noise and discontinuities, general and robust. Genetic Algorithms (GAs) have been suggested as a technique which meets these requirements (e.g. [17], [19]).

This section is intended to provide an introduction to Genetic Algorithms, concentrating on their abilities of function optimisation. Basic features of GAs are presented in the first part. The second part shows how GAs are employed to search images for instances of known objects.

Basic GAs' Features

As D. Goldberg states in his book [17], Genetic Algorithms are different from traditional techniques in four ways:

- "GAs work with a coding of the parameter set, not the parameters themselves." Parameters are encoded as bit strings (see Figure III-3). This encoding makes GAs independent of a particular problem and allows for employing principally the same technique to solve problems of different complexity. The bit strings are often called *chromosomes*, an analogy with the information-carrying objects in living cells (GAs draw much inspiration from the world of natural genetics). We will see later how to encode multiple parameters of a function.
- "GAs search from a population of points, not a single point." A population of N chromosomes is maintained. Since each chromosome represents one solution, we process

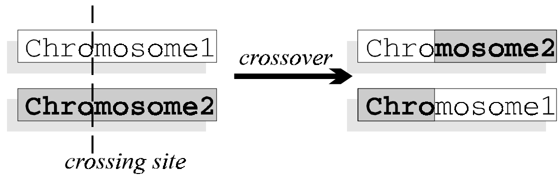


Figure III-4. Pair of chromosomes before and after crossover. Crossing site is chosen at random and parts of the chromosomes are exchanged.

N possible solutions at a time. This makes GAs more resistant to false (local) peaks of the objective function.

- "GAs use objective function information, not derivatives or other auxiliary knowledge." Relying only on the function value, the method gains generality. Unlike the classical calculus-based techniques, GAs do not require the existence of a gradient, and are therefore not limited to continuous or smooth functions.
- "GAs use probabilistic transition rules, not deterministic rules." Last but not least, this difference is inherent in the mechanism of creating a new generation. As mentioned before, GAs maintain a population of N chromosomes. The solution is obtained iteratively, imitating the behavior of a natural system, and creating new generations from the precedent using genetic operators *crossover* and *mutation*. Crossover takes two parent chromosomes (chosen from the parent population), chooses a random crossing site and swaps parts of the chromosomes (see Figure III-4). Mutation switches a bit of a chromosome. Which two chromosomes should mate and why these operators lead to a solution is explained below.

If we said before that probabilistic transition rules are used and the crossing site for crossover is chosen at random, we must distinguish between the classical brute-force random techniques, and the randomised operators of GAs. The latter work very well and yield good results thanks to the strategy called figuratively *selective breeding*. This means that the higher value of the objective function an individual had in the parent generation, the more descendants it will have in the next generation. Below average members will likely have less then the average number of children, and so bad solutions will gradually die off. Then, crossover ensures an exchange of information between the individuals, takes parts of the parents to combine them and create possibly fitter children. Mutation serves as an insurance against loss of information. Analogy of this process to natural selection is clear: the fittest members of a population mate to bear strong children.

Chromosome No.	Initial population	x value	f(x) = x^3	p_selection f/sum(f)	Actually selected
1	0 0 1 0 1	5	125	0,00	0
2	1 0 0 0 1	17	4913	0,17	1
3	1 1 0 0 0	24	13824	0,47	2
4	1 0 1 1 0	22	10648	0,36	1
Sum			29510	1,00	4
Average			7378	0,25	1
Max			**13824**	0,47	2

Table III-1. GAs at work. Initial population, objective function values and selection of chromosomes to form the next generation

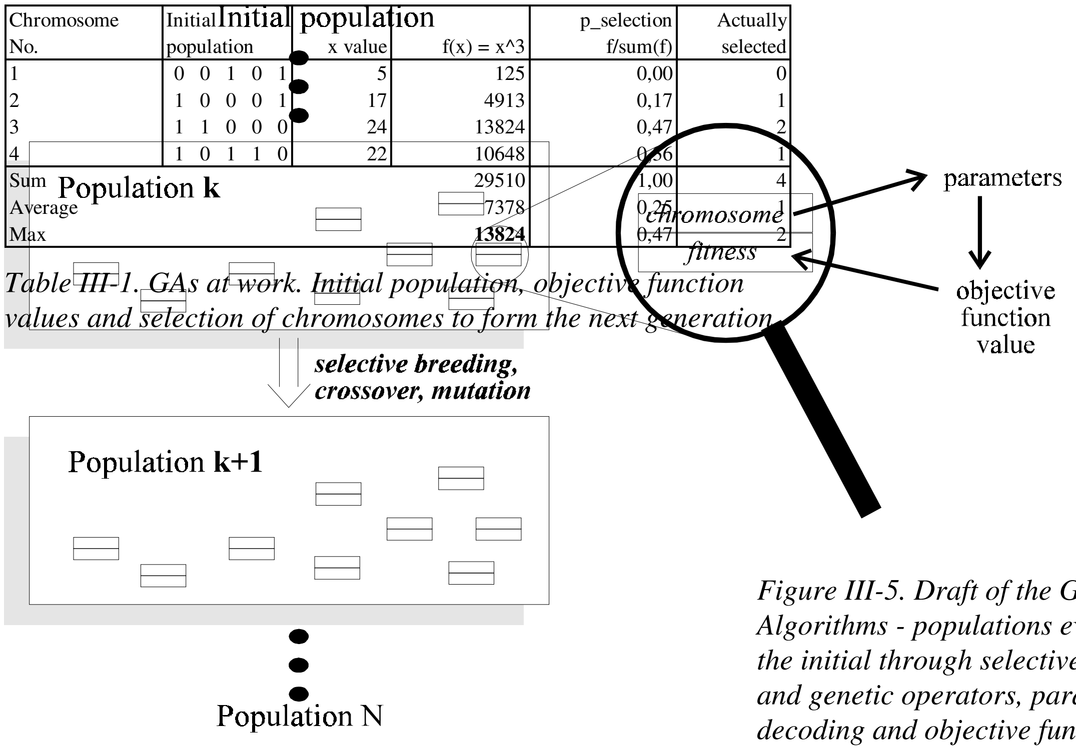


Figure III-5. Draft of the Genetic Algorithms - populations evolve from the initial through selective breeding and genetic operators, parameter decoding and objective function evaluation link the mechanism to a specific task.

GAs at work

Let's try a very simple simulation to clarify how GAs work. Imagine we have to optimise the function $f(x) = x^3$, for x being from the interval <0, 31>. First, we must choose the encoding of the parameter. Here the solution is straightforward, x will be stored as a five-bit unsigned integer.

The optimisation process starts with the initial population, randomly generated. At the beginning we simply fill the chromosomes (bit strings) with 0's and 1's at random. Then, to link the population to a specific problem, objective function values are computed for the chromosomes. Note that **chromosome decoding and the objective function are the only elements specific for a problem**. The rest of the mechanism stays generally the same for every task (see Figure III-5).

When each member of the population has its fitness (objective function) value assigned, we can move on to create the next generation. To do this, we have first to decide which individuals are going to take part in the process of creation. We employ the strategy of selective breeding in this way: probability of selection is computed for every chromosome based on its relative fitness value. Then a weighted coin toss is performed for each chromosome in turn. If the coin shows 'yes' (it does it with the corresponding probability of selection), the chromosome is reproduced into a group of prospective parents. The selection procedure is repeated until the size of the group is equal to the size of the generation. Best individuals are likely to have more than one copy in the group. Table III-1 gives an example: initial population, fitness values, probability of selection and actual number of selected chromosomes.

Having selected the parents for the following generation, crossover is the next step to come. Individuals mate randomly and crossover swaps parts of their bit strings to create a new generation. The result with the corresponding decoded parameters and the objective function values is shown in Table III-2.

If you compare the populations in Table III-1 and Table III-2, an important observation

Selected chromosomes	Mate (randomly	Crossover site selected)	New population	x value	f(x) = x^3
1 0 0 0 ⋮ 1	2	4	1 0 0 0 0	16	4096
1 1 0 0 ⋮ 0	1	4	1 1 0 0 1	25	15625
1 1 0 ⋮ 0 0	4	3	1 1 0 1 0	26	17576
1 0 1 ⋮ 1 0	3	3	1 0 1 0 0	20	8000
Sum					45297
Average					11324
Max					**17576**

Table III-2: GAs at work. Chromosomes selected for mating, crossover and the new population. (Note: probability of crossover was set to 1.0, probability of mutation was 0.05 and didn't occur.)

emerges: only by applying the selective breeding strategy and by swapping parts of the chromosomes, we were able to improve the maximum, as well as the average objective function values. This is the principle of Genetic Algorithms. Repeat this selection and mating N times for a population of a reasonable size, and you have a fair chance of obtaining the optimal result.

Why do GAs work so well?

How is it possible, that we should get something valuable only by combining parts of strings? Well, if we imagine that sub-strings of the chromosomes can represent solutions to sub-problems of our task, it becomes quite natural to couple these sub-strings to find a global solution. In [17] again, a quotation from mathematician J. Hadamard is found:

“... it is obvious that invention or discovery, be it in mathematics or anywhere else, takes place by combining ideas.”

But it is not only this feeling that somewhat human-like behaviour should lead to success, subtle mathematics lies in the foundations of Genetic Algorithms as well. Probably the most important notion is that of schemata and building blocks. Detailed explanation is beyond the scope of this article and can be found in the Goldberg’s book [17]. The principal idea is that the number of highly fit, short sub-strings (building blocks) will increase at least exponentially, because better individuals get more trials and short blocks are not likely to be cut by crossover (because the crossing site is chosen at random, short building blocks have a better chance to be missed and survive than longer strings).

Genetic Algorithms Applied to Image Search

When intending to employ GAs for model-based image search, we have to decide on encoding of the model parameters into a chromosome and to link the GAs to the task by a particular objective function.

Another note comparing our approach to that of Hill, Cootes et al.: we were inspired by their articles (e.g. [8]) to employ Genetic Algorithms for the global scope image search. All the rest, i.e. parameter coding, objective functions etc., has been developed directly from the GA textbook [17] or invented completely.

Parameter coding

As our model is parametrical, allowed to change position, scale and orientation, it comes naturally that all these parameters, to be stored in a single bit string, have to be encoded. The simplest (and usually successful) way goes via discretisation, mapping and concatenation.

Imagine you are facing the following situation: the input image, where the model is to be found, is of size 512x512 pixels, but you know (it was learned during training of the model) that the model centre can lie only in the interval [200, 300], for both x- and y- co-ordinates. You can choose to employ four bits for encoding of each of these position parameters. Then all we have to do is find the linear mapping which translates the four-bit unsigned integer (a natural expression of the bit string) into the desired interval [*min, max*]. The precision of this coding is calculated:

$$p = \frac{max - min}{2^l - 1}, \qquad \text{where } l \text{ is the length of the string.} \qquad \text{(III-1)}$$

Let a be the unsigned-integer evaluation of the bit string. The mapped decoding of a parameter follows:

$$m = min + a \cdot p\ .$$

In our case of 4 bits to encode the interval [200, 300], the precision is p=100/15=6.67 points, and an example, string 0101 (binary 5) translates into m=200+5*6.67=233.35.

Here, we started from the string length and computed the precision. We can obviously (and maybe more reasonably) go the other way - decide first what precision is required for the model parameters and determine how many bits are needed to store them. So we can set the position parameters to be represented with (minimal) precision 2 pixels, p=2. It follows from equation (III-1) above:

$$l = \frac{\ln\left(\frac{max - min}{p} + 1\right)}{\ln 2}$$

In our case, representing the position parameter in a bit string with precision 2 pixels needs l=ln(100/2+1)/ln2=5.67 bits, which has to be rounded up to the nearest integer, l=6 bits.

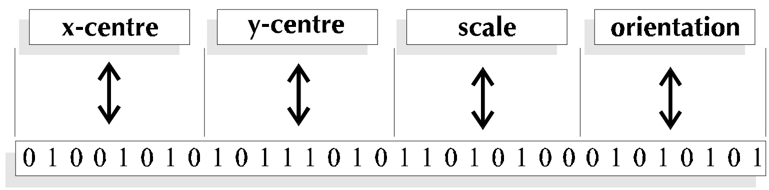


Figure III-6. The model parameters are coded and concatenated into a single bit string - chromosome.

We have shown how a single parameter can be stored in the bit string, but how should we cope with more parameters? The general rule suggests (see [17]) to keep relevant information together in short blocks. This can be translated into one possible solution: concatenate, i.e. join the individual parameters' strings into one chain. This way, all the parameters form together a single, longer bit string (Figure III-6).

We have to make two remarks: first, although the chromosome is formed by several parts corresponding to the parameters, this division plays absolutely no role when choosing the random position for crossover. Considerations of the string meaning are not necessary, even undesirable when mating two chromosomes. If we limited the crossing site to the boundaries between parameters and didn't allow the individual parameters to combine, we would restrict the parameter's value to those few randomly created in the initial population.

Second remark: the arrows in Figure III-6 point in both directions between the parameters and the chromosome to express mutual correspondence. However, the information flows only from the bit string to the decoded parameters during the Genetic Algorithms' work. We note it once more as a very important GA feature, even though we have to choose the chromosome length and we need a notion of parameter encoding to decide on the precision and number of bits, only the parameter *decoding* takes part during the GA process. We remind the reader of Figure III-5, exclusively decoding of the parameters with the objective function assignment link the general GA mechanism to a particular problem.

Fitness function

After parameter decoding, fitness (or objective) function serves to GAs as the second connection to a particular problem, and a connection of crucial importance. To help and encourage development of generations leading to a successful solution, objective function should provide reliable payoff information on individual chromosomes - solutions. In the domain of image search, the function will yield an evaluation of the quality of fit between the data and the model in a particular situation, given by the model parameters. Once more and hopefully for the last time, we have to mention the cycle *chromosome - decoded parameters - objective function - fitness - next generation chromosomes* etc.

The objective (fitness) function of the model projected into the image data is computed as a sum of fitnesses of individual model points. For each point, we look in its neighbourhood to find the location, where the appearance stored for the point best corresponds to the image. Then the fitness is directly proportional to quality of the fit (weight), and inversely proportional to the distance between the current and the best fit position (displacement). The algebraic form follows:

$$obj\_function = \frac{1}{N}\sum_{i=1}^{N} p\_fitness_i \ ,$$

$$\text{where} \qquad p\_fitness_i = weight_i * \left(const1 + \frac{const2}{1 + displacement_i} \right) \qquad \text{(III-2)}.$$

The weight of the fit was allowed to vary between 0 and 1, and we chose *const1*=30 and *const2*=70, to have the maximum possible fitness equal to 100. The member *const1* was placed into the expression after empirical experiments, so that the contribution of points, where good fit was encountered, is not so dramatically decreased by the displacement member. There is no one strict rule to obey when forming the fitness formula. We have experimented with another form, arising from the thought that we may omit the displacement factor completely - if a good fit is found within a reasonable distance from the point, the point will be allowed to move to a better position during a refinement step, we do not need to take the distance into consideration at this Genetic Algorithms rough solution stage:

Figure III-7. Linear scaling of the objective function under normal conditions.

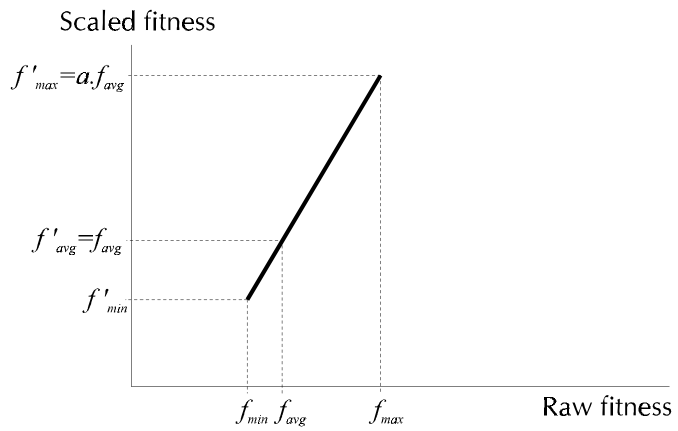
$$p\_fitness_i = weight_i * 100 \qquad \text{(III-3)}.$$

Apart from the generally slightly larger fitness values with the latter form, the two formulas (III-2) and (III-3) really lead to slightly different results. While the former prefers perfect fit of a part of the model points, the latter favours good fit along the whole boundary, even if individual points are slightly misplaced. We will point this difference out when speaking about results on real MR data. Interested reader is encouraged to look up another example of objective function in the article [18].

Fitness scaling

Having computed the objective function values from the image data (note that the objective function must be nonnegative - some mapping may be needed if it doesn't fulfil this requirement naturally), the fitness values return into the GA process and generations advance. Before the mating starts, we have to mention a concept, which is not necessarily contained in the simplest systems, but can contribute to a correct Genetic Algorithms behaviour - the concept of fitness scaling.

You may remember from the introduction and from the $\boldsymbol{x^3}$ example that selective breeding strategy is employed when creating a new generation. The higher is a member's relative fitness (i.e. his fitness / sum over the population), the more children it will have, good solutions are preferred. Two problematic situations can occur: at the beginning most of the population members are likely to have their fitness values very small, while few lucky chromosomes have it several times larger. Based on the fitness, only few individuals would mate and this could lead to a too early specialisation. On the contrary, after several generations, large part of the population will have similar fitness level and it will be difficult to choose the best. In other words, the same absolute differences between individuals represent excessive relative differences when the values are small, but small relative differences when the average fitness value rises. To avoid the early specialisation and to enforce competition at a later stage, fitness scaling serves usefully.



The principle of the method lies in the attempt to maintain the relative fitness of the fittest member to the average one at the same level during the whole course of the computations. One useful solution is linear scaling (see Figure III-7):

$$f' = a.f + b$$

where f and f' are respectively the original and the scaled fitnesses. The coefficients *a* and *b* are constrained to conserve the average fitness of the generation: $f'_{avg} = f_{avg}$. The scaling factor *a* can be usefully set between 1.2 and 2.0 for small populations (50-100 members). However, we must be careful and check that no members of the generation will be scaled to a negative value. The reader should consult the book [17] for a more involved discussion.

Experiments

In a GA image search experiment the model was created from 40 points to represent a rectangle with side ratio 2:3, of colour lighter than background. The scene dimensions were 512x512 pixels, the x and y co-ordinates of the model centre were limited to the interval <100, 400>. The scale of the model was allowed to vary between 0.6 and 1.5 of the original size, no constraints were put on rotation. We used 25 bits to encode these model parameters in a chromosome. Concerning the GAs' settings, the probability of crossover was set to 0.6, and the probability of mutation to 0.03. Size of the populations varied between 40 and 150 individuals, number of generations was limited to 50.

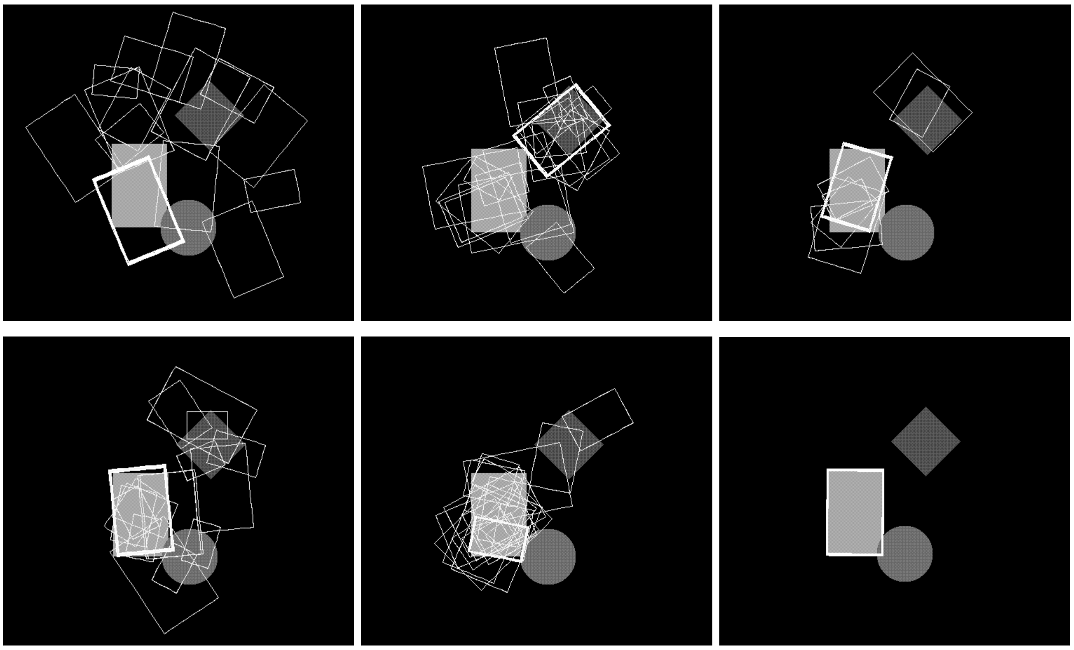


Figure III-8. Genetic Algorithms used for model-based image search. The objects in the scene are shown in grey, instances of the model in white. Bold white line shows the instance with highest fitness value of the generation. Pictures show various stages of the solution: top left - several members of the initial generation; top middle, top right, bottom left, bottom middle - above average individuals of generations 10, 20, 35 and 50, respectively; bottom right - final refined solution.

A typical run can be seen in Figure III-8. At the beginning (top left), individuals are spread randomly in the search space. As new generations are created, chromosomes are combined and combinations of parameters investigated. Gradually, more individuals concentrate in the area of main interest (i.e. of higher values of objective function). The bottom right picture shows the final result - the best approximate solution found over generations has been refined by a local technique to match optimally the image object. The model located the rectangle in the image data, even though the rectangle was partially occluded by a circle. More information on combining GAs with local techniques will be found in the following sections.

Note that the globally best solution, even if discovered in a generation, is not guaranteed to propagate through the generations to the final one. It may be easily destroyed by crossover or mutation. Compare pictures bottom left and middle - a very good estimate of the best solution appeared in the 35th generation, but didn't survive till generation 50. This has one advantage: the algorithms resist (possibly) false peaks of the fitness function and keep trying to locate a better solution. The remedy is simple, we have to store the best result over generations.

Figure III-9 shows an example of how maximum and average fitness values behave during the generations evolution for our image search problem. The maximum fitness jumps up and down as highly fit members are created and destroyed by crossover or mutation. The

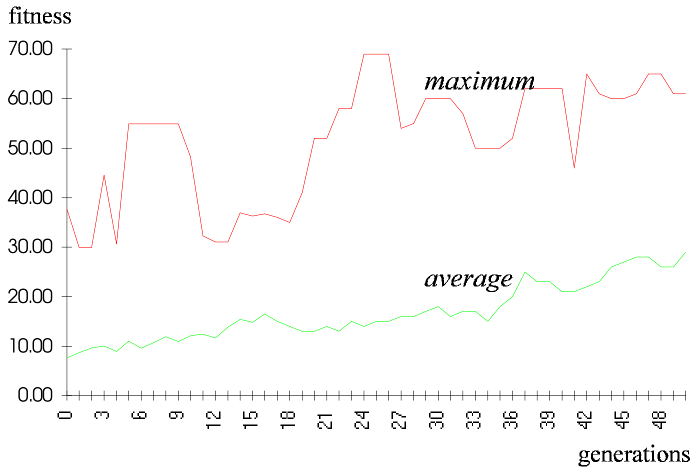


Figure III-9. Development of maximum and average fitness values for the image search problem. Each generation in this experiment consisted of 50 individuals.

average fitness value of the population grows steadily, and so pushes the maximum value up.

We have to say that Genetic Algorithms are not guaranteed to find the optimal result (obviously, as no method not checking *all* possible parameter combinations in a large search space can be). The success ratio depends on the length of the chromosomes (i.e. size of the state space), on size of the GA population, on number of generations investigated, on the Genetic Algorithms settings - probabilities of crossover and mutation etc., but it depends also on type and complexity of the input scene.

We have conducted another experiment with a shape model of „heart“ and the scene of Figure III-10. As seen from the pictures, the model tries to match any encountered object as well as possible, and it appears that both the fruit shapes are quite similar to that of the model.

Varying the size of the populations for our heart search problem between 50 and 200 individuals with the number of iterations limited to 200, we obtained the success ratio between 50 (for the smallest) and 75 percent (for generation of size 150). The bigger the population, the higher the probability of success should be; however, there is obviously some limit of what is reasonable. The smallest population provides only poor reliability, but yields the results very quickly, in few generations. We tried to compute the number of brute-

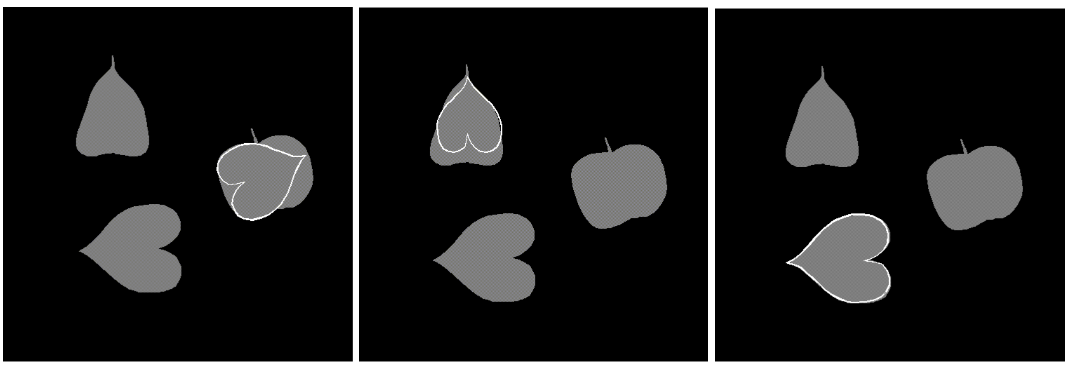


Figure III-10. The heart experiment - the scene consists of drawings of an apple, a pear and a heart. Successful search result on the right.

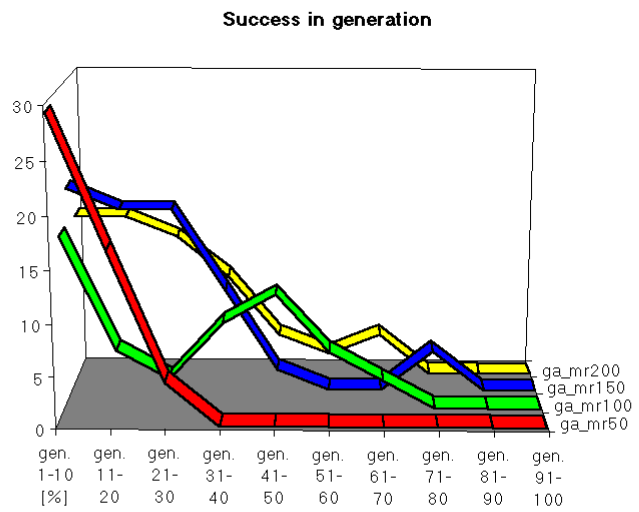


Figure III-11. Probability of obtaining the correct solution grouped in respect to the actual generation where this success is achieved. (Axes meaning: vertical - success [%], left-to-right - generation number, front-to-back - population size)

force random trials, which would be necessary to obtain the same probability of success with the same precision as the Genetic Algorithms do. We found empirically the displacement which the local refinement technique (from the next section) is able to accept and still achieve the correct solution. We computed that the GAs were required to set correctly at least 17 from the 27 bits of the chromosome employed in this experiment. With the population of size 50, eight generations, i.e. 50x8=400 trials were needed on average to achieve the 50% success. To obtain the same result, 162 times more brute-force trials would be necessary. The most reliable 150 members' run spent on average 22 generations to acquire success ratio of 72 percent, which is more than 28 times faster compared to the brute-force method.

On the other hand, adding more generations doesn't show much influence as mature populations contribute only a little or nothing to the overall success ratio.

Figure III-11 presents the measurements taken during the heart experiment: all the successful results were obtained before generation 80 for all population sizes. For the smallest generation the observation is much more severe - no more success was attained after generation 30. The conclusion could be not to continue after generation 50 if a satisfactory result has not been found yet, rather restart the computation from the beginning.

III. 2.2. Local refinement of the model parameters

One significant feature of the Genetic Algorithms is that they base the optimisation uniquely on the objective function and do not employ derivatives or other supplementary knowledge. This leads to generality of the approach and makes it suitable even for discontinuous and noisy functions, but imposes unpleasant limitations on the method's accuracy. If the objective function can be considered as piece-wise continuous, we may use the gradient information profitably. The Genetic Algorithms yield an approximate solution which is then refined by a local technique. This combination is often described figuratively: GAs find a hill of the fitness function, and a local technique goes and climbs it. In [8] and related articles, such a local refinement technique is dubbed *Active Shape Models* (ASM).

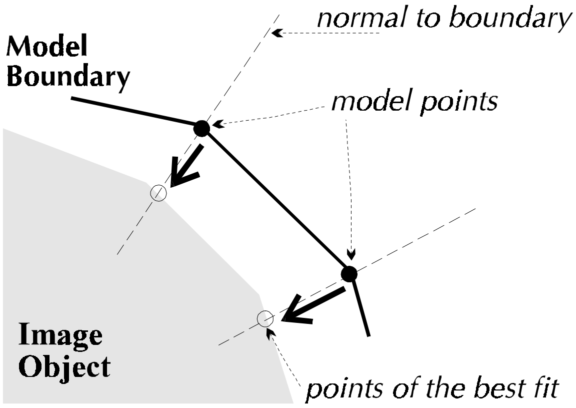


Figure III-12. First step of the local refinement: each model point compares its stored appearance to the image data along the normal to boundary and finds the location of the best fit. Bold arrows signify suggested movement of the model points.

The first step of this method is visualised in Figure III-12. A line of specified length is scanned from the image for each model point; the line is normal to the model boundary and centre of the line is positioned at that point. Then the appearance stored for the model point (mind you, it can be either an edge or a grey-level profile) is compared to the line of image data and the point where these two show minimal difference is found. From these best fit points, a *set of suggested new locations* of the model points is computed (it can be e.g. half the way between the old and the best fit position to allow convergence).

Let $\boldsymbol{X}_1 = (x_{10}, y_{10}; x_{11}, y_{11}; \ldots x_{1n-1}, y_{1n-1})^T$ be the set of original model points, $\boldsymbol{X}_2$ analogous set of suggested points from the paragraph above. We would like to map $\boldsymbol{X}_1$ onto $\boldsymbol{X}_2$ so that the distances of all corresponding points are as small as possible. Formulated mathematically, we have to find such parameters (translation, rotation and scale) which applied to $\boldsymbol{X}_1$ transform it onto $\boldsymbol{X}_2$ optimally in the least square sense. We mark $\boldsymbol{X'}_1$ the set of transformed $\boldsymbol{X}_1$ points and can write the function to be minimised:

$$f = \sum_{i=0}^{n-1} w_i \left[\left(x'_{1i} - x_{2i} \right)^2 + \left(y'_{1i} - y_{2i} \right)^2 \right] \qquad \text{(III-4)}$$

The points transformed from the original model points are expressed:

$$x'_{1i} = s.\mathbf{cos}\,\varphi.x_{1i} - s.\mathbf{sin}\,\varphi.y_{1i} + t_x \qquad \text{(III-5)}$$

$$y'_{1i} = s.\mathbf{sin}\,\varphi.x_{1i} + s.\mathbf{cos}\,\varphi.y_{1i} + t_y$$

where scale s, rotation φ, and translations t_x and t_y are the unknown to be found.

We merge the equations (III-5) into the prescription for function f (III-4) and, following a corrected suggestion of [13], substitute:

$$a = s.\mathbf{cos}\,\varphi \qquad \text{and} \qquad b = s.\mathbf{sin}\,\varphi\,.$$

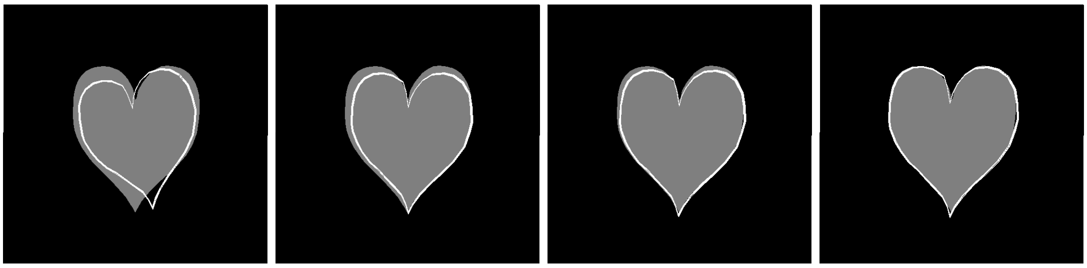


Figure III-13. Local refinement of the heart model's parameters. The initial position on the left, followed by the fifth, tenth and fifteenth (final) iterations.

The set of variables for which the function f is minimal can be found by the least-square approach. We set each of the partial derivatives of f with respect to variables a, b, t_x and t_y equal to zero. This leads to a set of linear equations (written in matrix form):

$$\begin{bmatrix} Z & 0 & X_1 & Y_1 \\ 0 & Z & -Y_1 & X_1 \\ X_1 & -Y_1 & W & 0 \\ Y_1 & X_1 & 0 & W \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \\ X_2 \\ Y_2 \end{bmatrix}$$

where $X_k = \sum_{i=0}^{n-1} w_i \cdot x_{ki}$ $\qquad Y_k = \sum_{i=0}^{n-1} w_i \cdot y_{ki}$

$$Z = \sum_{i=0}^{n-1} w_i \cdot \left(x_{1i}^2 + y_{1i}^2\right) \qquad W = \sum_{i=0}^{n-1} w_i$$

$$C_1 = \sum_{i=0}^{n-1} w_i \cdot \left(x_{1i} \cdot x_{2i} + y_{1i} \cdot y_{2i}\right) \quad \text{and} \quad C_2 = \sum_{i=0}^{n-1} w_i \cdot \left(x_{1i} \cdot y_{2i} - y_{1i} \cdot x_{2i}\right)$$

This set of equations can be solved by standard matrix methods and the variables s and φ extracted from a and b. To summarise, we have completed a numerical method able to yield the parameters translation, rotation and scale which map one 2D shape onto another optimally in the least square sense, and it does so by a single solution of 4 linear equations of 4 variables.

Let us return back to the context of our local refinement technique. We have obtained the suggested new locations of the model points and computed new model parameters to match the model to them rigidly. It is probable that some deformations were present in the suggested points, and the next step would be trying to cover these deformations by the deformation parameters included in the model. However (because we have not allowed deformations of our model at this stage), we are not going to explain this in detail and refer the reader to [11] or [8].

The iterative cycle *model points - suggested points - new model parameters - new model points* etc. is repeated until the model parameters become stable. All the final results of the experiments above were improved by this local technique. Figure III-13 shows the refinement process in detail.

III. 2.3. Combining Genetic Algorithms and local refinement

We have said that a hybrid technique, combining the Genetic Algorithms (for generation of hypothesis and approximate solutions) with a local technique (refining the hypothesis) has proved useful. But how to merge these two approaches together?

The simplest possibility would be to run GAs for a specified number of generations, take the best member of the last generation and then employ the local refinement on it. We already know that best solutions are easily destroyed during the GA development, so the best member of the last population would not mean necessarily the best hypothesis ever, and this combination would not show a good success ratio. Still keeping the structure of Figure III-14 left but allowing the dotted lines, we could improve the performance by storing the best

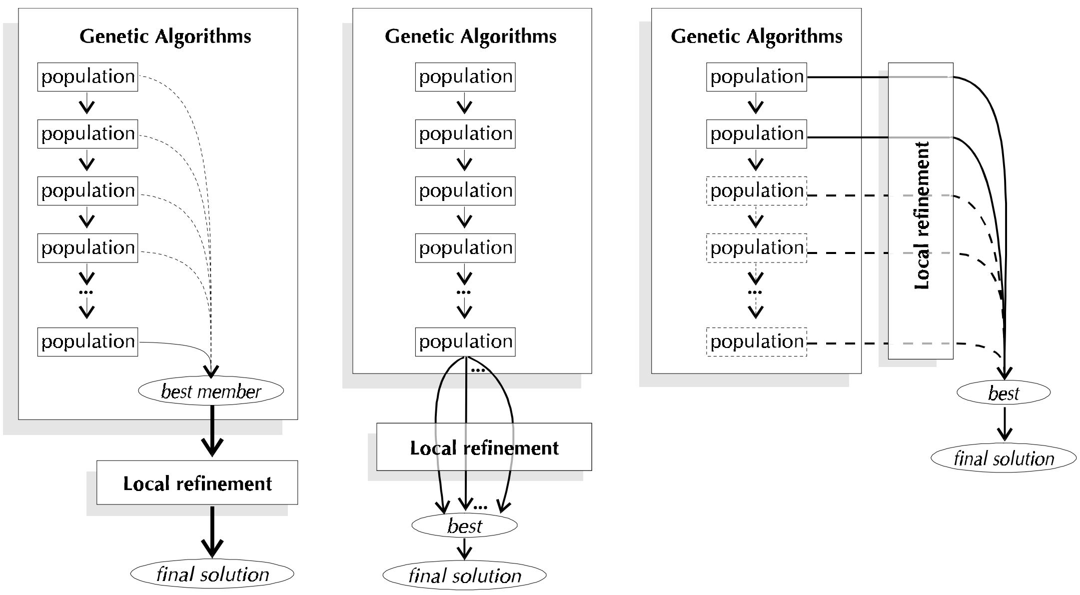


Figure III-14. Various combinations of Genetic Algorithms and local refinement. The method sketched on the right, refining the best individual of each generation and possibly finishing if the solution obtained so far is good enough,appears to supply the best results.

GA solution over generations.

Different technique: we may choose to enforce the GAs to yield multiple different solutions by allowing *sharing* of the fitness function (see the Goldberg's book [17] for explanation), then take several of them, refine and select the best (this approach is drawn in the centre of Figure III-14).

The last possibility, and it has shown to be actually the most successful, is to refine the best GA population member of each generation created and reserve the best refined solution over generations (observe Figure III-14 right). This generation-by-generation refinement has two important advantages: none of the good (i.e. best in a generation) GA hypothesis is lost, and we can stop the computation after a small number of generations if a good solution is found. Obviously, this statement on early finish (which should have been symbolised by the dotted lines in Figure III-14 right) is valid only if we are able to distinguish what is a „good solution". In our experiments, where the maximum of the objective function is set to 100, the stopping limit of fitness greater then 50 seems restrictive enough, but this half-of-the-maximum rule cannot be generalised.

III. 3. Application to MR data

To prove the method's ability to work successfully on MR data, we created a model representing the brain ventricles. This model in the original slice (it is slice number 133 from the MR dataset) is seen in the centre of Figure III-15, where it delineates the dark area in the centre. This initial position was drawn manually in the data, the model is formed by 40 points. Grey-level profiles of length 7 pixels were extracted from the image and stored as appearance for each of the model points.

Figure III-15. Model of ventricles - position in the original slice.

Then we tried to employ this model to find ventricles in the neighbouring slices. The results are presented in Figure III-16. We can see that the model provides a good approximation of the desired solution in the slices adjacent to the original one. Even though placed rigidly, the boundary is not far from the contours seen in the image, and could be easily refined by a snake-like technique.

On the other hand, our 2D rigid model is not suitable for segmentation of the further slices, where the shape differs more from the original. To be able to process them reliably, we would need another 2D model, or preferably a fully three-dimensional model, which would represent and extract the 3D object as a whole, not as a set of slices. Such a model is dealt with in the chapter to come, we will stick to our 2D experiment for a moment.

A surprising experience: we created the GA fitness function so that it would yield the fitness values in the interval 0-100. And it really did so, the successful final solutions of the experiments on artificial data presented earlier showed fitness values between 75 and 98 (percent). No doubt we cannot expect it to be 100 percent (caused by inaccuracies related to the discrete pixel grid and transformations of the model), but the results on real MR data

Figure III-16. The ventricle model applied to neighbouring slices of the original one. Top, from left to right: slice -6, -3 and -1, bottom: slice +1 and +3, all relatively to the original slice.

were much lower. For example, the value was only 33 for the correct solution in the original slice, and about 20 in the adjacent images. It seems that the grey-level profiles are very sensitive to even a small displacement and related change of the data scanned from the image.

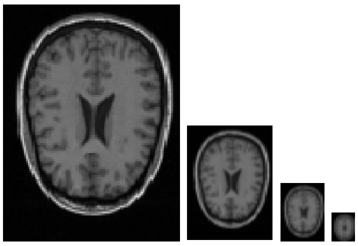
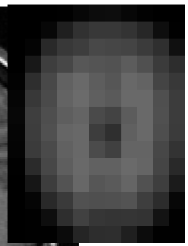



Figure III-18. Multiple resolution levels of the input image. On the left, subsampled versions of the Figure III-15 data. On the right, the image of resolution level 4 (the smallest on the left) magnified to 800%.

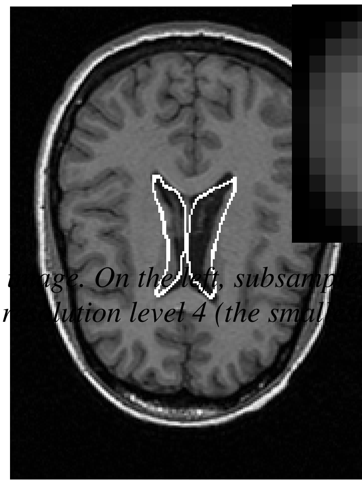


Figure III-17. Serch result in the slice original -3 using the fitness of equation (III-3).

All the figures above were meant for the form (III-2) of the objective function. The form (III-3) yields results slightly different. First, the maximal obtained fitness values proved to be higher (as supposed from the mathematical prescription) and varied between 30 and 55 for the searches in the three images most similar to the original one. Second, more important difference can be seen in Figure III-17. It shows search result in the same image as was the middle in Figure III-16 top, and visualises practical importance of a good fitness function. Whereas the former (i.e. the (III-2) equation and corresponding Figure III-16) attracts the model too strongly to a perfect fit of a part of the model boundary, the latter (equation (III-3)) encourages correct global solution along the whole length of the boundary by ignoring the displacements of the individual model points. Although simpler in form, the prescription (III-3) provided a better result in this case.

Anyway, despite the low fitness levels (even the maximum 55 for the simpler fitness function and the original data is not very much) and the resulting difficulties in evaluating good and unsatisfactory solutions, the shape model proved to be very stable and fast. All the results shown in the previous page were obtained by the first run of the program and in one of the first few GA generations. A great deal of the robustness is due to an approach we have not mentioned yet, *multiple resolution search*.

The principle is simple and is employed frequently in many domains: instead of working with the whole amount of input data, we take only a rough version, we sub-sample the input image. This way, we reduce dramatically the size of the data and also omit the finest details. An example is shown in Figure III-18. If you compare it to its counterpart in Figure III-15, you can observe the effect of subsampling practically. The smallest image has been magnified for you to see the difference; it is 16 times smaller in both directions, i.e. the amount of memory needed to store it is already 256 times inferior to that of the original.

We build such multi-resolution pyramid both when creating the model, so the grey-level profiles are trained for each of the resolutions separately, and when processing the input data. Then the Genetic Algorithms search is performed on the coarsest level only, and the local refinement technique runs from this coarsest down to the finest resolution, effecting several, e.g. ten iterations at each resolution level. Aside from the reduced amount of data, the main importance of such an approach lies in improved robustness. The coarse-level profiles, being of the same pixel length as the fine and working on subsampled data, cover a much larger part of the image and are less likely to be caught by nearby, misleading image features.

The performance of the method rises significantly as well. This is allowed by the fact that longer jumps of the model points (and parameters) are possible at the coarser levels, and thus the model faster reaches convergence. Second, *and this is a novelty compared to the article of Cootes and Taylor [10] where a similar method appeared*, we employ two times less points at each step of the resolution. Stability of the iterations is not crucified as the points would overlap anyway, and the reduction of computational cost is even more significant.

IV. Three dimensions

We are going to introduce an extension of the 2-D shape model presented in the previous chapter into three dimensions. New features are discussed in detail and compared to those of two dimensions.

IV. 1. Model representation

Similarly to the 2D case, model shape is represented by n connected points forming the object boundary, only this time three co-ordinates are needed for a single point:

$$\mathbf{x} = (x_0, y_0, z_0; x_1, y_1, z_1; \ldots x_{n-1}, y_{n-1}, z_{n-1})^T,$$

where (x_i, y_i, z_i) is the i^{th} point of the shape. As well as in 2D, each labelled point represents a particular position on the object.

The reader who has not skipped the previous chapter knows what is signified by the different typeset. Before we continue, we have to make a very important comparison of our method and the 3D approach to MR data segmentation presented by A. Hill and his colleagues in [20]. Their method is an extension of the 2D Point Distribution Model and it retains the basic features: the 3D shape model is created through training and analysis from a set of examples. Plausible deformations are stored within the model representation as main modes of variations. Their so called 3D model has one serious restriction though: it is not fully 3D, rather 2.5-dimensional. This criticism is based on two observations: first, the model is created from 2D contours (extracted from MR data slices) by forming generalised cylinders, i.e. connecting the contours in adjacent slides - see Figure IV-1. Only these up-down links are allowed and where a more complex shape is encountered, it must be divided into several sub-parts. Second objection, and here the two-dimensional origins of the method are clearly visible, changes to the model orientation are possible only in one plane, about

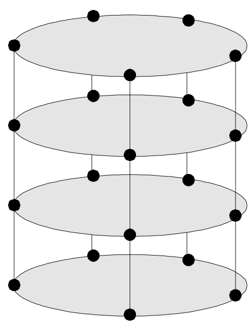


Figure IV-1. The Hill's 3D model consists of cylinders formed by connected 2D contours.

the z-axis, normal to the 2D MR slices. All other 3D rotations are captured as the model's modes of variation, i.e. as deformations and displacements of the individual planes.

However, after these offensive notes on terminology, we must confess that their technique seems to provide better results than any other method previously seen. After all, raw MR data are obtained as a set of slices. But still, we have tried not to cast such restrictions on our approach and went in the direction of a clearly and completely 3D model.

Back from England to our method: we allow any connections of the model points. Typically, the model will be obtained from (segmented) 3D volume through triangulation of the surface. This way the obtained model points are not restricted to lie on several „floors“ of the original MR slices and the resulting mesh covers the surface uniformly.

As well as in 2D, the model appearance is represented by grey-level profiles, normal to object boundary and centred at the model points (see Figure II-5).

IV. 2. Model geometry

Probably the most important difference between the 2D and 3D model is due to the model geometry. While only four parameters (x- and y-translation, scale and one for rotation) suffice for capturing all rigid model movements in two dimensions, the number rises to seven in 3D. These parameters include three for translations (along the x, y and z axes), one for scale and three for rotations. Extension of translation and scale into 3D is straightforward, but the three parameters of rotation require further explanation. Before we get down to it, we will try to refresh the reader’s notion of transformation matrices.

IV. 2.1. Transformation matrices

As mentioned in any better elementary textbook on computer graphics (such as the Czech Žára’s [47]), co-ordinate transformations can be usefully performed by means of transformations matrices. In 3D, if every point (x, y, z) is expressed by four co-ordinates as $P = [x_h, y_h, z_h, 1]$, we can write the point’s transformation in matrix form:

$$P' = P.\mathbf{M} ,$$

where **M** is the 4x4 transformation matrix. If the point should undergo more transformations, the individual transformation matrices may be multiplied first (this means the transformations are compounded into one) and then only one multiplication is performed on the point.

As we are concerned with rigid transformations uniquely, the resulting matrix may be formed only by multiplication of the following:

Translation by a vector (T_x, T_y, T_z)

$$\mathbf{M}_T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

scaling by factor s

$$\mathbf{M}_S = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

... and rotations, which are not so straightforward and deserve a separate section.

IV. 2.2. Rotations in 3D

When thinking of parametrising rotations in 3D, what typically comes first to mind are the historically most popular Euler angles. All possible orientations of an object must be achieved by three successive rotations about the three co-ordinate axes, called by Watt & Watt in [45] (a book which this section is based upon) figuratively rolls.

The rotation about each of the axes can be written in form of the transformation matrix (corresponding to Figure IV-2, from top to bottom):

$$x\_roll(\theta_1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_1 & \sin\theta_1 & 0 \\ 0 & -\sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$y\_roll(\theta_2) = \begin{bmatrix} \cos\theta_2 & 0 & -\sin\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_2 & 0 & \cos\theta_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$z\_roll(\theta_3) = \begin{bmatrix} \cos\theta_3 & \sin\theta_3 & 0 & 0 \\ -\sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

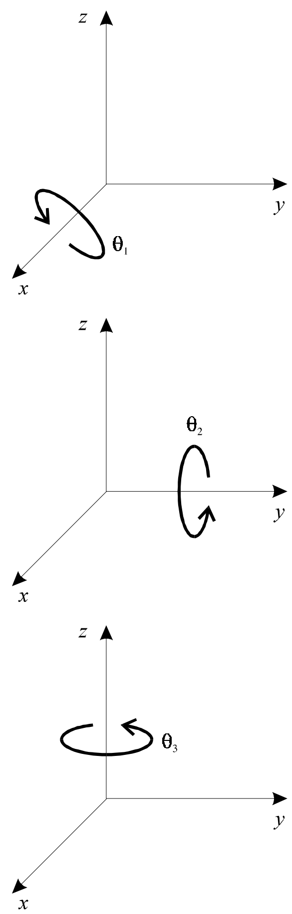


Figure IV-2. Euler angles - all possible rotations described as a composition of rotations around the three co-ordinate axes.

A compound rotation $R(\theta_1, \theta_2, \theta_3)$ about the three axes is acquired by multiplication of these matrices.

Several serious problems are inherent in Euler angles. As multiplication of matrices and the equivalent rotations are not commutative, the three parameters for x-, y- and z-rolls can form 6 different rotations depending on the order in which they are applied. Vice versa, having a compound rotation matrix, the task of extracting the individual roll parameters is under-constrained because of the mutual dependence of the rotations. While one compound transformation matrix represents uniquely one rotational transformation (understood physically, having two positions of an object), it can be decomposed into several sets of the Euler angles.

The meaning of the three angles' composition is unclear either: their interdependence disables any judgement on the rotation's importance, i.e. we cannot say how much the object rotated between the initial and the final position. This notion of the „size“ of the rotation is very important when intending to restrict the possible orientations of an object to some range (and this is the case: we build an anatomical model and require its rotations to be easily controllable). Can we hope to find a parametrisation of rotation which would allow such a disposition ? Probably yes, and this optimism originates in the Euler theorem which tells us that **it is possible to get from one orientation to any other by a simple steady rotation about a single axis**. This single rotation could be treated much easier than the three successive rolls.

Angular displacement

Still pursuing the reasoning of [45], this notation follows from the Euler's theorem. We can define any rotation as angular displacement given by $\boldsymbol{R}(\theta, \boldsymbol{n})$ of an amount θ about an axis $\boldsymbol{n}$. Let's mark A a point in 3D and A' the same point after rotation $\boldsymbol{R}(\theta, \boldsymbol{n})$ has been applied on it (see Figure IV-3). Furthermore, we select an arbitrary point on the axis $\boldsymbol{n}$ and mark $\boldsymbol{r}$ the vector connecting this point to A. By applying the rotation $\boldsymbol{R}$ on this vector, it translates into $\boldsymbol{Rr}$, joining the point on the axis with the rotated A'. We refer the reader to [45] again for the derivation, which is accomplished quite simply by decomposing $\boldsymbol{r}$ into the parts parallel and perpendicular to the axis $\boldsymbol{n}$, and mention only the result: the rotated vector can be written as

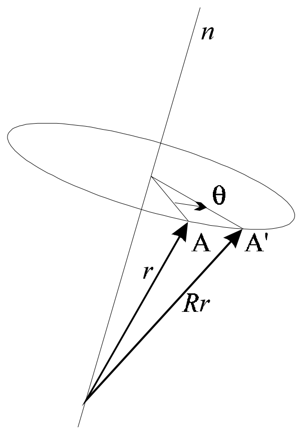


Figure IV-3. Angular displacement R(θ, n) of point A and vector r.

$$\boldsymbol{Rr} = (\cos\theta)r + (1 - \cos\theta)n(n \cdot r) + (\sin\theta)n \times r \tag{IV-1}$$

We will see in a moment how this equation relates to the desired parametrisation of orientation.

Quaternions

A quaternion is a number $q = a + b\boldsymbol{i} + c\boldsymbol{j} + d\boldsymbol{k}$, where $\boldsymbol{i}$, $\boldsymbol{j}$, and $\boldsymbol{k}$ are three imaginary units. The basic idea is similar to, or rather extended from that of complex numbers[8]. The three imaginary units have the following properties:

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = -1 \qquad \boldsymbol{ij} = \boldsymbol{k} \qquad \boldsymbol{ji} = -\boldsymbol{k}$$

with the cyclic permutation $\boldsymbol{i} \rightarrow \boldsymbol{j} \rightarrow \boldsymbol{k} \rightarrow \boldsymbol{i}$.

The authors of [45] introduce a condensed notation:

$$q = (s, v), \qquad \text{where } (s, v) = s + v_x\boldsymbol{i} + v_y\boldsymbol{j} + v_z\boldsymbol{k}.$$

s is thought of as the scalar part of the quaternion and v the vector part with axes $\boldsymbol{i}$, $\boldsymbol{j}$, and $\boldsymbol{k}$.

From the above rules properties of quaternions can be easily derived, such as formulas for multiplication of quaternions etc. But here comes the reason why this all is mentioned here.

We take a pure quaternion (i.e. with no scalar part) $p = (0, r)$

and a unit quaternion $q = (s, v)$,

and define $R_q(p) = q.p.q^{-1}$.

This expands (also by rewriting the unit quaternion $q = (\mathbf{cos}\,\theta, n.\mathbf{sin}\,\theta)$) to

$$R_q(p) = (0, \mathbf{cos}(2\theta)\,.r + (1 - \mathbf{cos}(2\theta))\,n\,(n.r) + \mathbf{sin}(2\theta)\,.(n \times r))$$

Now compare this to the equation (IV-1). The two are identical in form except for the factor 2 appearing in the angle. This means no less than the following: rotating a vector r by an angular displacement (θ, n) is the same as taking this angular displacement, lifting it into quaternion space by representing it as a unit quaternion $(\cos(\theta / 2), \sin(\theta / 2)\,.\,n)$ and performing the operation $q(\ )q^{-1}$ on the quaternion $(0, r)$.

In other words, we can parametrise rotation in terms of the four parameters

$$(\cos(\theta/2), \sin(\theta/2)\,.\,n_x\,, \sin(\theta/2)\,.\,n_y\,, \sin(\theta/2)\,.\,n_z)$$

using quaternion algebra to manipulate the components.

[8] A very nice story is connected with the invention of quaternions, we cannot omit it when copying the rest of the reasoning from [45]. Sir William Hamilton, a great mathematician, had been interested in complex numbers since early 1830's. Complex numbers have the form a+bi and define a plane, where one axis is real and one imaginary. For over 10 years Hamilton tried to extend this concept in order to define a volume by adding one more imaginary axis. However, he could not succeed. Then, on 16 October 1843, when walking past Broome Bridge in Dublin towards the Royal Irish Academy, he suddenly realised that three rather then two imaginary units were needed, with the properties defined in the text above. He was so excited that he carved the formulae for this newly discovered and named *quaternion* on the side of the bridge.

Chosen parametrisation of the model orientation

We cease following the Watt & Watt's book strictly for a while and present a thought of our own here. This parametrisation above is redundant, as the vector n is required to be of unit magnitude and thus its end must lie on the unit sphere. But to parametrise all points on the sphere (i.e. all the possible vectors n which once more means all the axes of rotation of the object) spherical polar co-ordinates may be employed profitably, needing two instead of three parameters to describe the axis orientation. Furthermore, as a rotation of θ about an axis $\boldsymbol{n}$ is identical to the rotation $-\theta$ about the axis $-\boldsymbol{n}$ (see Figure IV-4), we do not need to cover but a half of the unit sphere.

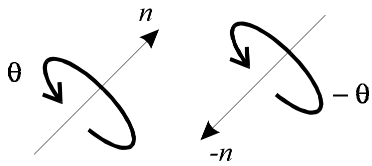


Figure IV-4. Identical rotations about axes n and -n.

To summarise, we use three parameters to express all the possible rotations of an object (Figure IV-5). By chance, all the three are angles, but different from the Euler angles criticised above. Two of them (α and β) are understood as spherical polar co-ordinates and serve for describing the axis of rotation (i.e. the three numbers n_x, n_y and n_z for the quaternion notation are computed from them). The third angle, θ, then expresses actual magnitude and direction of the rotation about this axis. This magnitude of the rotation can be easily limited to some interval centred at zero.

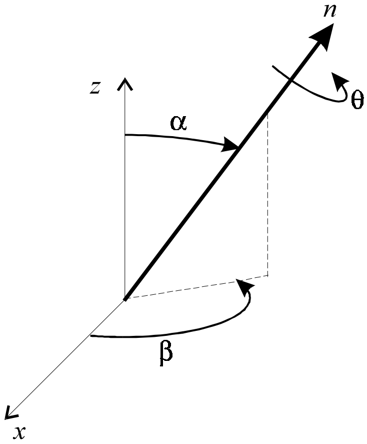


Figure IV-5. Rotation parametrisation in terms of three angles: α and β (spherical polar angles) define the axis n, while θ states the actual rotation about that axis.

However, two problems still persist in the axis parametrisation: first, the spherical polar co-ordinates do not cover the space uniformly, being, similarly to the meridians on the Earth, more dense around the poles. We could take this into consideration when thinking about more frequent or preferred types of rotations. Second, it could be rather complicated to restrict the acceptable axes of rotation to some subset of all possibilities, i.e. specify only some areas on the sphere where the axis is allowed to pass.

Moving into and out of quaternion space

We have decided to parametrise model orientation by an axis and an angle of rotation about that axis, and have shown that quaternions may be employed helpfully as a mathematical representation thereof. But when intending to apply this orientation onto the individual model points, it would be an extremely perverse way to lift every point to the quaternion space, perform the operation, convert it back to three-dimensions and complete the rest of the transformations, i.e. scaling and translations there. Instead, we should leave the points where they are, and manipulate them traditionally by transformation matrices (this has been proved the computationally cheapest way - once we have the transformation matrix complete having composed all the matrices of individual transformations, we only multiply all the points by the same matrix). To be able to do so, we need a mechanism translating the rotations from the quaternion form into the rotation matrix and vice versa. Fortunately, such a conversion exists and has been published in the Watt & Watt's book [45].

From a quaternion to a transformation matrix

We have the rotation in the quaternion form

$$(\cos(\theta/2), \sin(\theta/2) \,.\, n_x \,, \sin(\theta/2) \,.\, n_y \,, \sin(\theta/2) \,.\, n_z).$$

For reasons of simplicity, we write it as (*W, X, Y, Z*). Then the rotation matrix corresponding to this quaternion-described rotation follows:

$$\begin{bmatrix} 1-2Y^2-2Z^2 & 2XY-2WZ & 2XZ+2WY & 0 \\ 2XY+2WZ & 1-2X^2-2Z^2 & 2YZ-2WX & 0 \\ 2XZ-2WY & 2YZ+2WX & 1-2X^2-2Y^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(IV-2)}$$

From a rotation matrix to a quaternion

Having the general rotation matrix

$$\begin{bmatrix} M_{00} & M_{01} & M_{02} & 0 \\ M_{10} & M_{11} & M_{12} & 0 \\ M_{20} & M_{21} & M_{22} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(IV-3)}$$

the only problem is to convert it into the form (IV-2) directly above. Having obtained the parameters *X, Y, Z* and *W*, we have found the quaternion (*W, X, Y, Z*) and can easily compute the angle θ with the axis *n.*

It follows from the matrix form (IV-2) and the fact that the rotational quaternion must be of unit magnitude, that (see [45] again)

$$W = \left(\frac{1}{4} \sum_{i=0}^{3} M_{ii} \right)^{1/2}$$

If W is not equal to zero, the other three quaternion components can be computed using these:

$$X = \frac{M_{21} - M_{12}}{4W} \qquad Y = \frac{M_{02} - M_{20}}{4W} \qquad Z = \frac{M_{10} - M_{01}}{4W}$$

If W is equal to zero, these equations are undefined and other combinations of the matrix components must be investigated to calculate *X*, *Y*, and *Z*. We refer the reader, for the last time, to [45] for a complete listing of the conversion algorithm in C.

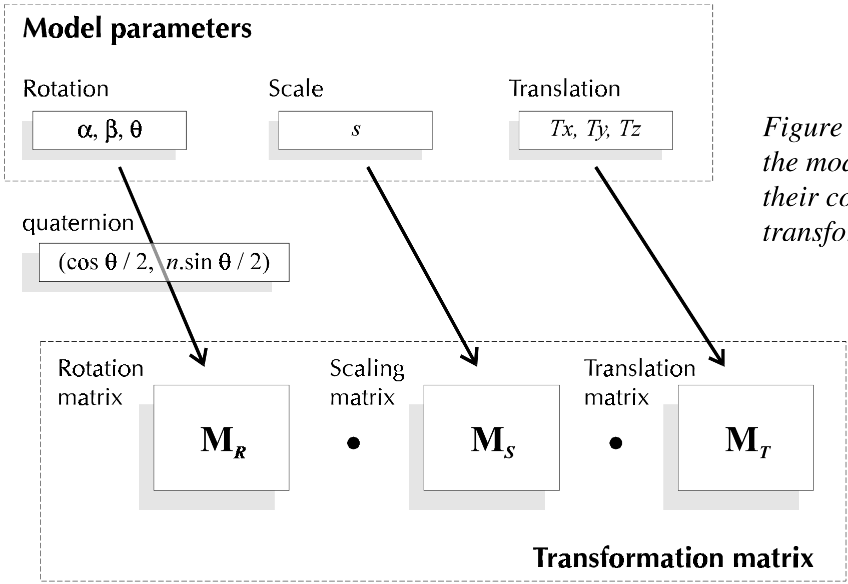


Figure IV-6. Overview of the model parameters and their conversion into the transformation matrix.

IV. 2.3. Model geometry summarised

Not to let the reader lose the context and the long continuous thread of ideas running hopefully through the whole book, we summarise the distributed notes on the model geometry here.

Our 3-D model needs **seven parameters** to express all the feasible, rigidly acquired states. **Three for rotation**: two of them determine an axis and the third gives the actual rotation about this axis; **one** parameter **for scaling**; **three** parameters **for translations** in the directions of the three co-ordinate axes x, y and z.

When transforming the model from its initial or „normal“ position, we perform only one multiplication of each of the model points by a transformation matrix **M**. This matrix has been acquired by multiplication of the individual matrices as symbolised in Figure IV-6. In the initial positions the model is centred at the origin of the co-ordinate system, so we can rotate the model about an axis going through the origin without the risk of displacing the model centre. After the rotation (or before it, as the order of these two operations makes no difference), scaling is applied. At last, the scaled and rotated model is shifted according to the three translation parameters.

The reader might be curious to know why the conversion of rotation parameters into a rotation matrix goes via the quaternion space, while the other two are performed directly. There are several reasons for it: unlike the other two, the rotational transition is not so straightforward; the transformation into the quaternion is easy and the conversion from a quaternion to the matrix, which is computationally inexpensive, was known at the moment we were looking for something similar; and last, we behave like mathematicians at this point[9].

---

[9] A very nice quiz appeared in the Smullyan’s book [49]: are you rather a mathematician or a physicist? Quoted somewhat inaccurately, since we don’t have the original version available, it follows.

Imagine you have an empty kettle, cold water, cooker, teapot and tea leaves, and you are asked to make tea. What do you do?

IV. 3. 3-D data search

When intending to employ the 3-D shape model for model-based data segmentation, the procedure is basically the same as in the 2-D case. First, Genetic Algorithms, a global optimisation method, are applied to provide an approximation, then a local technique slightly adjusts the model parameters to better match the data.

As for Genetic Algorithms (GA), the process remains unchanged. Even the original 2D fitness function, linking the general GA mechanism to the particular problem, can be adopted without modifications, still based on the sum of the quality of the model-to-data fit of individual model points. The only obvious difference lies in the parameter coding, in that having more parameters longer chromosomes will be needed to store them. Longer chromosomes also imply larger GA populations and probably more generations necessary to achieve a good solution.

On the other hand, the situation gets much more complicated with the local technique of parameter refinement. We will investigate it in detail.

IV. 3.1. Parameter refinement

The first step of parameter refinement stays the same as in 2D. Figure III-12 showed the idea - each of the model points looks in its neighbourhood and suggests the location where it would like to move in order to increase the fit between the appearance stored for the point and the image data.

The second step, although of the same objective as that in two dimensions, i.e. find such model parameters which would map the model optimally onto these suggested points, this second step is qualitatively much more complicated then in 2D. Not only there are more parameters, but the relations between these parameters are more complex too. Notoriously, the most serious problems are caused by rotations. We spent a lot of time seeking after the equations which would serve for this computation, but where we were able to apply the least-squares approach in 2D, differentiate the transformations and after a simple substitution acquire a set of linear equations for the model parameters, in 3D we obtained only long confusing formulas not leading to any solution.

Fortunately, we were given a valuable advice and a reference to a set of articles dealing with the problem of 3D transformation fitting ([1], [42], [23]) and can summarise the method here.

Did you answer you fill the kettle, heat the water, put tea leaves into the teapot and, when the water boils, pour it into the teapot? All right. Now imagine a different task: the situation is exactly as above, you have the kettle, tea etc., only now the kettle is full and the water is boiling. How do you make tea now?

Did you answer you just put tea leaves into the teapot and pour the boiling water onto it? Of course it's correct, and you are rather a physicist. A mathematician would empty the kettle in the sink and thus convert the problem into the previous, already solved.

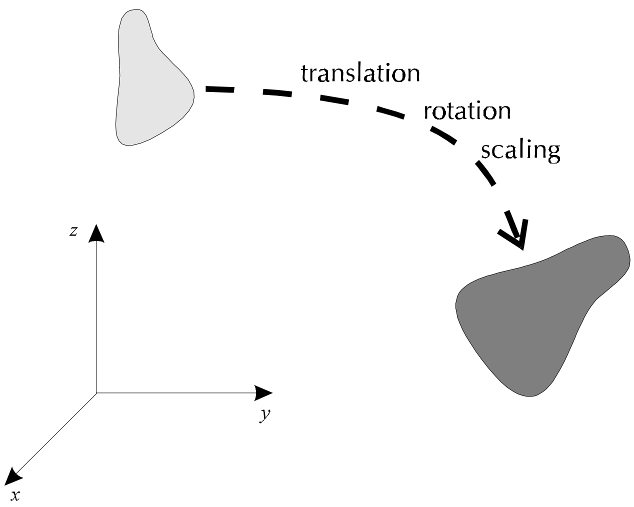


Figure IV-7. Sketch of the 3D transformation fitting problem. We have two objects (a model and a set of suggested points) and want to find the parameters which map one onto the other.

We start with the situation sketched in Figure IV-7. We have two clouds of N 3-D points each (one of which represents the model and the other the set of suggested points), we know the corresponding pairs, i.e. where should each point be mapped.

Translation

First, we compute the **centres of gravity** of both point sets, and shift the sets in space so that they are centred at the origin of the co-ordinate system. Having completed this, both rotations and scaling with respect to the origin (which means the usual case) can be applied freely without the risk of introducing unwelcome displacement of the object's centre. Also, the three **parameters of translation** from one to the second position are computed directly from these two centres of mass.

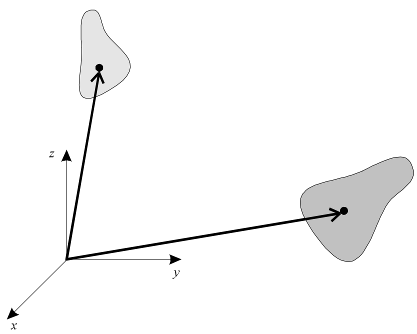


Figure IV-8. Translation parameters are computed as the difference between the two centres of gravity.

Rotation

Next comes the crucial task, the nub of the matter, we calculate the rotations. We follow the Kanatani's article [23] and mark **R** the rotations, i.e. a matrix 3x3 formed by the first three rows and first three columns of the general rotation matrix (IV-3). Now, given the two sets of points (centred at co-ordinate origin) $\{\mathbf{x}_i\}$ and $\{\mathbf{x'}_i\}$, i = 1,...,N, where the i-th point has three co-ordinates (x, y, z), we want to find the rotation which maps one set onto another optimally in the least-squares sense, i.e. we want to minimise the term

$$\sum_{i=1}^{N} w_i \left|\mathbf{x_i} - \mathbf{Rx'_i}\right|^2 \qquad \text{(IV-4)}$$

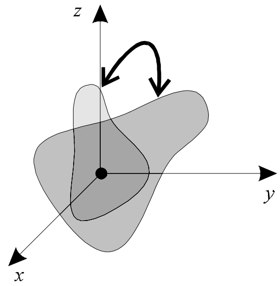


Figure IV-9. Rotation is calculated for the objects centred at co-ordinate origin.

where w_i is a positive weight for the i-th pair of points. Note that we keep the notation of [23], dashed vectors (e.g. **x'**) here signify the model, while the plain **x** stands for the set of suggested points.

The *correlation matrix* **K** between $\{\mathbf{x}_i\}$ and $\{\mathbf{x'}_i\}$, i = 1,...,N, is defined by

$$\mathbf{K} = \sum_{i=1}^{N} w_i \mathbf{x_i} \mathbf{x'}_\mathbf{i}^T .$$

Here and henceforth, the operator ^{T} signifies vector or matrix transpose.

One way of solving our minimisation problem goes via decomposition of the covariance matrix **K** into the form:

$$\mathbf{K} = \mathbf{V}\Lambda\mathbf{U}^T, \quad \Lambda = \begin{pmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{pmatrix}, \quad \sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$$

where **V** and **U** are orthogonal matrices. This decomposition, called *singular value decomposition,* belongs to standard methods of linear algebra. A description of both the underlying theory and the procedure how to obtain it from the matrix **K** can be found e.g. in [33].

Kanatani proves in [23] that if $\mathbf{K} = \mathbf{V}\Lambda\mathbf{U}^T$ is the singular value decomposition, the term (IV-4) is minimised over all rotations by

$$\mathbf{R} = \mathbf{V}\begin{pmatrix} 1 & & \\ & 1 & \\ & & \mathbf{det}\left(\mathbf{V}\mathbf{U}^T\right) \end{pmatrix}\mathbf{U}^T \qquad \text{(IV-5).}$$

To summarise in simple words: we find the rotation matrix which matches the two sets of points centred at origin by computing their covariance matrix, finding singular value decomposition of the matrix, and introducing the results into the equation (IV-5).

Now we have the rotation matrix, but this itself does not provide a clear, real-world idea of what exactly the rotation is like. We would like to measure (and to constrain) how much actually our model rotates. Here again, it is the right domain for quaternions.

We have already shown the procedure performing the conversion from a rotation matrix to a quaternion. We employ it here and acquire the rotation in the quaternion form

$$(\cos(\theta / 2), \sin(\theta / 2) \,.\, n_x \,, \sin(\theta / 2) \,.\, n_y \,, \sin(\theta / 2) \,.\, n_z).$$

From this we can simply move to our three rotation parameters (α, β, θ - if you are not certain about their meanings, look back at Figure IV-5) by standard goniometric computations, especially when we have choosen that the axis n is allowed to point to only a half of the all possible directions (once more, if you ask how we can dare restrict it to a half of the unit sphere only, consult the section on our model rotation parametrisation above) and know that e.g. n_z is always greater or equal to zero.

Scale

Last and not complicated, we compute the remaining scale parameter as a ratio of (weighted) average distances of the two sets' points from the co-ordinate origin (note the two sets are still centred there). We mark **d**(p) the distance of point p from the co-ordinate origin and can write:

$$s = \frac{\sum_{i=1}^{N} w_i\, \mathbf{d}(\mathbf{x}_\mathbf{i})}{\sum_{i=1}^{N} w_i\, \mathbf{d}(\mathbf{x'}_\mathbf{i})}$$

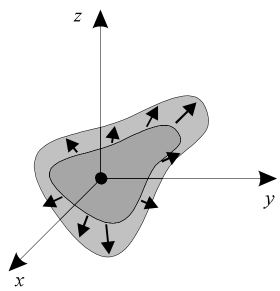


Figure IV-10. Scale for the fit is obtained through the average distance from the centre.

It makes no difference whether we compute the scale before or after applying the rotation.

IV. 3.2. 3-D object fitting experiment

To check or prove the method's ability to compute correctly the model parameters which map one set of points onto another, we performed several experiments, fitting artificially created objects, such as the two of Figure IV-11. The two models, which should symbolise an apple and a pear, are represented by connected model points on the objects' boundaries. In this image the models lie at the locations where the lines cross. We know what are the mutually corresponding points on both objects (both these entities consist of the same number of points).

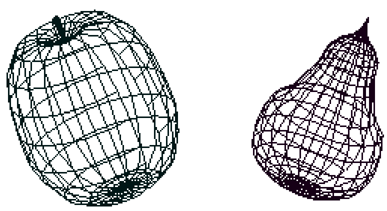


Figure IV-11. Two objects of the transformation fitting experiment.

We took the apple model and transformed it into several positions using various translation and rotation parameters. Then we ran the computations to find the transformation parameters, which would map the second object onto the first in these positions optimally. As can be seen in Figure IV-12, where the transformed apple and the pear with the calculated parameters are shown together, the procedure finds the transformations to fit the two objects reliably, although the two shapes to be matched are relatively different.

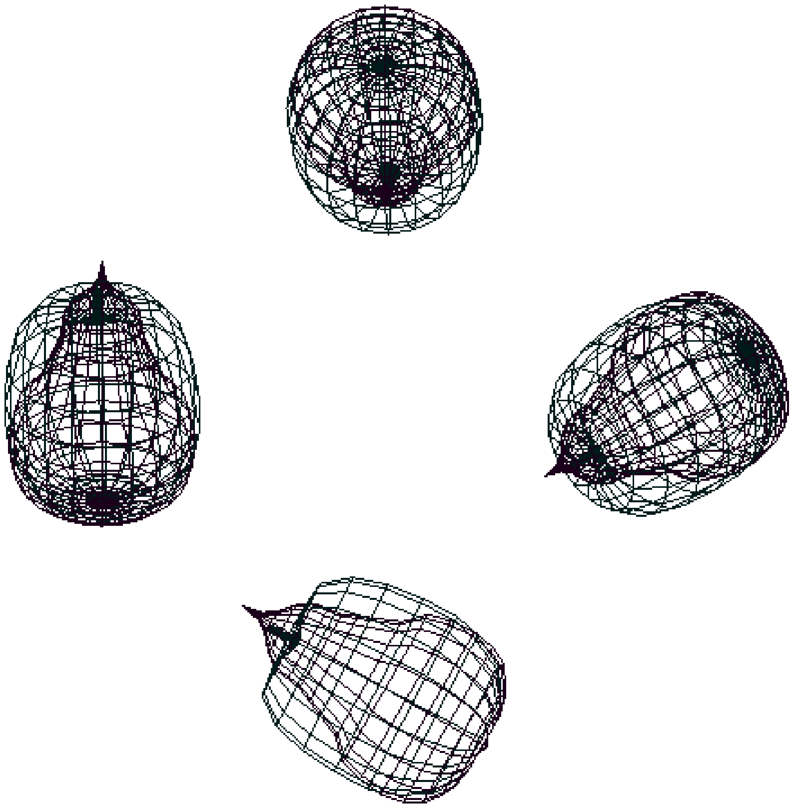


Figure IV-12. Three dimensional transformation fitting: the transformations for the pear model were computed to match the apple model optimally in the least-squares sense.

V. Implementation

We have implemented all the methods described in the previous chapters in the C++ programming language and have compiled them on a Silicon Graphics workstation. We are well aware of the fact that the system of objects (in the sense of object-oriented programming) and their methods, although approaching 200 kB of source code at the moment, is far from being complete and ready to use. However, we may briefly mention the structure of the program to supply the reader with a general idea on how the system has been designed. We believe that parts of the code, i.e. are individual objects or their functional groups, can be adapted and implanted easily (and we hope usefully too) in another system dealing with similar problems. Only basic ideas of the highest, global level are presented here, the reader is encouraged to seek more information on a particular function directly in the C++ source code.

V. 1. Model representation

The class **Model**, contained conjointly in the files „model.h" and „model.cpp", embraces data structures and methods needed to store and manage the representation of the object to be found in the data. Like most of the classes in the system, the general parent class Model has two children specific for the two-dimensional or the three-dimensional case (see Figure V-1).

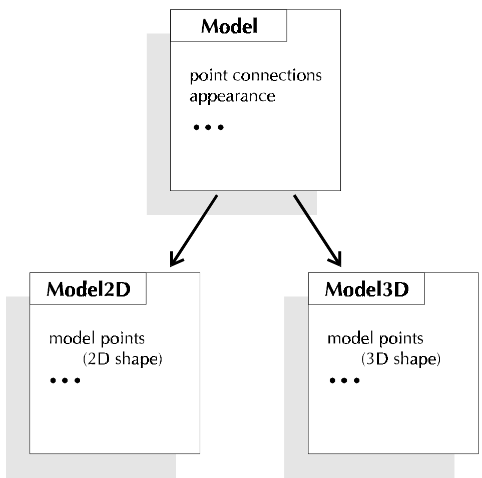


Figure V-1. Virtual parent class Model unifies the 2D and 3D specific children.

Co-ordinates of the model points in the normalised position, information on the point connections, grey-level profiles or other sort of appearance at the model points, acceptable intervals for the model parameters (e.g. feasible positions of the model centre, limits of scale and rotation), all these are contained in the Model class. Information about possible deformations of the

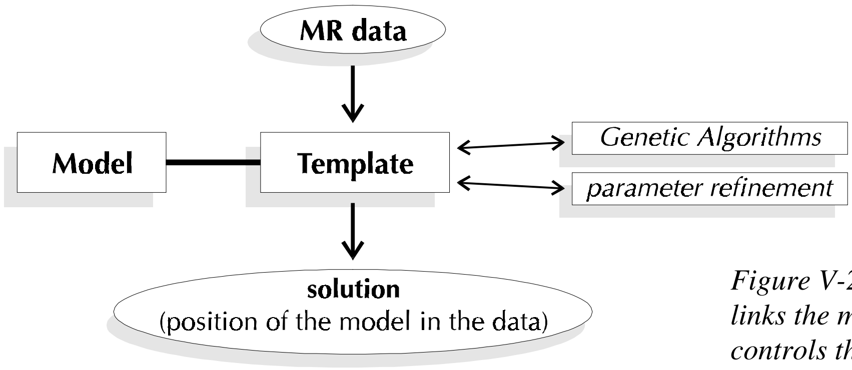


Figure V-2. Class Template links the model to the data, controls the segmentation process and provides solution.

model can be stored here as well, although this has not been used much yet.

The class member functions are oriented on manipulating of the data, which includes model import from and export to files, model formation and training of the model appearance from a set of points in an image or in MR data.

V. 2. Instances of the model

Class **Template** (again, with its 2D and 3D children) serves as an interface and a links the model to a particular task. The Template object must be created for the Model in to employ it for model-based 2D image or 3D data search.

The Template class contains, together with the necessary reference to the model and reference to the input data, parameters specifying the actual position of the model instance within the data, transformed points' co-ordinates, weights and locations of suggested points.

As far as the role of this class in the program context is concerned, we may say the Template class is the highest one, as it controls the whole segmentation process, employing the services of many other classes and linking the model knowledge to the data. Figure V-2 presents a sketch of the whole process from the „class" viewpoint: the Template, created for a Model, receives the input MR data, calls the global Genetic Algorithms search, refines the hypotheses obtained, and returns the best final solution.

V. 3. Input data manager

Expressed somewhat strongly, the heading intended to say that class **Bitmap** and its 2D and 3D instances are in charge of storing the input data (either MR data for the medical applications, or regular images) and of making their use easy for other objects. This is, for example, where the pyramid of multiple resolution data is maintained and access to it provided transparently to the user (the public co-ordinates of a point remain the same at all resolutions), here the set of 2D Magnetic Resonance slices is transformed into the 3D cube shown on the first page of this text. The class Bitmap also embodies functions for import and export of the data from and to graphic formats.

V. 4. Genetic Algorithms

The **GeneticAlgorithms** class is probably the easiest to be applied to a problem different from the image search it has been working on here. Genetic Algorithms are in principle a general optimisation technique, and this generality of the method was carefully maintained during the implementation. The code for the method is designed to be fully independent of the particular problem, the only link to a specific task being the address of the objective function, which assigns a fitness value to a chromosome. It is obvious that the objective function cannot operate without decoding the chromosome into some parameters, but this task-dependent process remains to be implemented elsewhere.

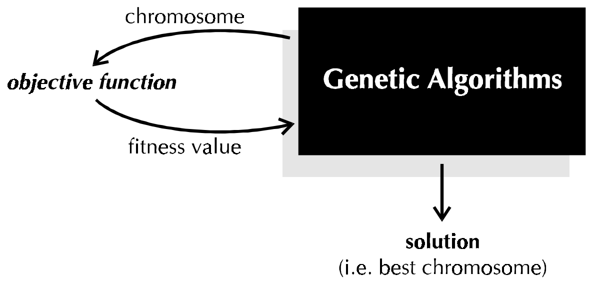


Figure V-3. The Genetic Algorithms class can be employed for generally any function optimisation without re-implementing. The „black box" is linked to the particular problem only by the fitness function call.

V. 5. Mathematical library

Last, but certainly not least, the files „mat3.h" and „mat3.cpp" assemble the grounds of other classes' functions and of their data structures. The general classes (some of them as templates) like Vector, Matrix or Point (2D and 3D) have been implemented to allow a higher level of abstractness in the more complex computations. The mathematical functions include those from the domain of linear algebra, 2D and 3D geometry or geometrical transformations. Operator overloading provides an extremely comfortable interface and makes these functions simple to invoke.

VI. Conclusion

VI. 1. Recapitulation

To summarise briefly what has been presented in the previous chapters: after a short introduction to medical examination Magnetic Resonance method and several notes on the character of the data it provides, we reviewed in detail the approaches to medical data segmentation which were published in several recent years. The rest of the book then discusses thoroughly the method we have developed and implemented.

Segmentation of medical data, and the Magnetic Resonance (MR) data represent a typical example, is impossible to accomplish without an external source of knowledge, which could help overcome the data unreliability. Information on anatomical shape of the object to be found serves as an example of such anatomical knowledge. This work concentrates on one possible representation of shape and appearance of anatomical structures. We store the information in form of a geometrical model, where both the shape of the object and the knowledge on its appearance are expressed by points forming the model surface. The model we have implemented is similar to that of Cootes, Hill and Taylor ([8], [20]), with the main differences being the 3D model geometry and transformations (we manage fully three-dimensional representation and rotations, while the other approach represented 3D object as a set of 2D contours) and the attitude to model deformations (we do not allow deformations at this stage).

The model-based segmentation is achieved by placing the model in the data (in various locations and with different parameters) and comparing the fit of the actual position to the data. Two distinct methods are applied to find the best match: first Genetic Algorithms, a global optimisation technique, provide an approximation, and then a local procedure refines the model boundary position to improve the solution. Both these methods are presented in detail in the chapter on 2D image search, together with numerous experiments on artificial images as well as on real MR data.

A chapter on tree-dimensional model is included, concentrating on the main difference compared to the 2D situation, i.e. the model geometry. As mentioned above, we allow full 3D transformations. The most problematic one, the rotation, is governed by the mathematical apparatus of quaternions.

Toward the end of the book, a short chapter mentioning several aspects of the C++ implementation can be found.

VI. 2. Future directions

Despite all effort, the method has not been completed yet. The first part to be finished is the implementation of the 3D model. We have already debugged most of the functions, so now only several details in the code are missing. A more serious problem, and we will come back to it in a moment, are the training and testing 3D data.

Second aspect to be dealt with are the model deformations. We intend to employ the rigid solution as an initial position for a snake-like refinement procedure. We would like to combine the traditional edge-seeking snakes and balloons with the powerful grey-level profiles as the appearance representation at the boundary points. We intend to measure how much of the model variability can be covered by this posterior deformation. However, we do not deny that inter-object variability represented directly in the model and admitted during the whole segmentation process should perform better. These acceptable deformations could be extracted from a training set of objects, or formed as a combination of the trained and artificially created variation modes (Cootes and Taylor presented a combined 2D model in [12]).

But all these extensions are qualified by a solution of the most serious problem associated with our medical data processing project: the absolute lack of medical experts taking part in it. Medical experts are needed urgently, both to help create the system, i.e. to perform the manual segmentation of the training data and to insert other sorts of anatomical knowledge in the system, and to provide the demand for the system so making the work on it useful and distinctly oriented. Without the perspective of a practical application, further development of the system remains a maybe interesting, but still a waste of time. We have tried to contact several doctors, but they were either too busy (or at least pretended to be), or were not willing to co-operate, not seeing a possibility how to employ the system in their particular situation. This way we are only wandering about, spellbound to a vicious circle: not having enough training data, we cannot create a perfect system, and not having a perfect application to show, we cannot persuade the doctors of the usefulness of the project and acquire more training data from them.

Epilogue & Acknowledgements

„… Everything's just as it was ! "
„Of course it is," said the Queen: „what would you have it ? "
*„Well, in **our** country," said Alice, still panting a little, „you'd generally get to somewhere else - if you ran very fast for a long time, as we've been doing."*
*„A slow sort of country ! " said the Queen. „Now, **here**, you see, it takes all the running **you** can do, to keep in the same place. If you want to get somewhere else, you must run at least twice as fast as that ! "*

Lewis Carroll in [5]

However unlikely it might seem, as I've been running as fast as *I* can, the thesis has reached its end. I don't dare say whether it got us somewhere or *everything's just as it was*. Anyway, I believe it has been an extremely useful experience for me to work on it, for this great task I have been striving to tackle forced me to learn a lot, both from the viewpoint of mathematics or computer science, and from that of presentation of scientific ideas and research results, either in form of this thesis or the previous conference articles. Now the show is over, the curtain has fallen and the actors are coming to take a bow.

From all those many who deserve my respect and gratitude I mention these few particularly involved in the project:

Ing. Jiří Žára, CSc.
who was at the beginning, assigned the work to me, and is from a great deal responsible for the inspiring ambience at the Computer Graphics Laboratory

Ing. Petr Felkel
my thesis supervisor, who had worked in the domain before I started to, introduced me into the field and has been always ready for assistance

Ing. Bedřich Beneš
who stimulated my first glance at the exciting world of quaternions

Ing. Tomáš Pajdla
whose advice helped inestimably with the 3-D point fit computations

Ruda Lecjaks
who has the degree Ing. today as well but did not when he helped me implement graphical output at the SGI workstations

&
all my friends and relatives
who either kept distracting me or let (and helped) me work on this project (depending both on time and person), thus permitting me to finish it without „allowing the work to rob me of my courage and my wits"

(quotation adapted from Castiglione's [51])

I would like to present my thanks to all of them. There may be another person who merits my respect: the unknown reader, who has been patient and tolerant enough to survive all the chapters I included in this book. If only he found it a little useful!

To conclude and to augment a little this work's literary quality, I will cite an author incomparably greater than I am, William Shakespeare:

POLONIUS: What do you read, my lord?
HAMLET: Words, words, words.

I have enjoyed gathering them into this volume (well, at least sometimes) ...

Pavel Mrázek
Prague, May 1996

References

[1] K. S. Arun, T. S. Huang, S. D. Blostein: Least-Squares Fitting of Two 3-D Point Sets. IEEE Trans. PAMI, Vol. 9, No. 5, September 1987

[2] D. R. Bailes, C. J. Taylor: The Use of Symmetry Chords for Expressing Grey Level Constraints. Proc. British Machine Vision Conference, Springer-Verlag 1992

[3] H. H. Barrett, A. F. Gmitro (Eds.): Information Processing in Medical Imaging, 13th International Conference, IPMI '93, Flagstaff, Arizona, USA, Proceedings. Springer-Verlag, Lecture Notes in Computer Science 1993

[4] Bomans M., Hoehne K. H., Tiede U., Riemer M.: 3-D Segmentation of MR Images of the Head for 3-D Display. IEEE Trans. on Medical Imaging, Vol. 9, No. 2, June 1990

[5] Lewis Carroll: Through the Looking-Glass and What Alice Found There. Wordsworth Editions 1993.

[6] Cohen L.D., Cohen I.: Finite-Element Methods for Active Contour Models and Balloons for 2-D and 3-D Images. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 15, No. 11, Nov. 1993

[7] D. L. Collins, A. C. Evans, C. Holmes, T. M. Peters: Automatic 3D Segmentation of Neuro-anatonical Structures from MRI. Y. Bizais et al. (eds.): Information Processing in Medical Imaging, Kluwer Academic Publishers 1995

[8] Cootes, Hill, Taylor, Haslam: The Use of Active Shape Models For Locating Structures in Medical Images. In: IPMI'93 (Reference [3])

[9] T. F. Cootes, C. J. Taylor: Active Shape Model Search using Local Gray-Level Models : A Quantitative Evaluation. Proc. British Machine Vision Conference, BMVA Press 1993

[10] T. F. Cootes, C. J. Taylor, A. Lanitis: Active Shape Models : Evaluation of a Multi-Resolution Method for Improving Image Search. Proc. British Machine Vision Conference, BMVA Press 1994

[11] T. F. Cootes, C. J. Taylor: Active Shape Models - 'Smart Snakes'. Proc. British Machine Vision Conference, Springer-Verlag 1992

[12] T. F. Cootes, C. J. Taylor: Combining Point Distribution Models with Shape Models Based on Finite Element Analysis. Proc. British Machine Vision Conference, BMVA Press 1994

[13] T. F. Cootes, C. J. Taylor, D. H. Cooper, J. Graham: Training Models of Shape from Sets of Examples. Proc. British Machine Vision Conference, Springer-Verlag 1992

[14] J. Declerck, G. Subsol, et el.: Automatic Retrieval of Anatomical Structures in 3D Medical Images. INRIA rapport de recherche No. 2485, Février 1995

[15] J. Declerck, G. Subsol, Thirion, Ayache: Automatic Retrieval of Anatomical Structures in 3D Medical Images. N. Ayache (ed.): Computer Vision, Virtual Reality and Robotics in Medicine, First Int. Conf., CVRMed'95, Nice

[16] K. Delibasis, P. E. Undrill: Anatomical object recognition using deformable geometric models. Image and Vision Computing Vol. 12, No. 7, Sept. 1994

[17] D. Goldberg: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley 1989

[18] A. Hill, C. J. Taylor, T. Cootes: Object Recognition by Flexible Template Matching using Genetic Algorithms. Proc. European Conference on Computer Vision (G. Sandini ed.), pp.852-856, Springer-Verlag 1992

[19] A. Hill, C. J. Taylor: Model-Based Image Interpretation Using Genetic Algorithms. Proc. British Machine Vision Conference, Springer-Verlag 1991

[20] A. Hill, A. Thornham, C. J. Taylor: Model-Based Interpretation of 3D Medical Images. Proc. British Machine Vision Conference, BMVA Press 1993

[21] A. Hill, C. J. Taylor: Automatic Landmark Generation for Point Distribution Models. Proc. British Machine Vision Conference, BMVA Press 1994

[22] V. Hlaváč, M. Šonka: Počítačové vidění. Grada 1992.

[23] K. Kanatani: Analysis of 3-D Rotation Fitting. IEEE Trans. PAMI, Vol. 16, No. 5, May 1994

[24] T. Kapur, Grimson, Kikinis: Segmentation of Brain Tissue from MR Images. N. Ayache (ed.): Computer Vision, Virtual Reality and Robotics in Medicine, First Int. Conf., CVRMed'95, Nice

[25] M. Kass, A. Witkin, D. Terzopoulos, Snakes: Active contour models. Int. J. Comput. Vision, vol. 1, 1987.

[26] J.-F. Mangin, F. Tupin et al.: Deformable Topological Models for Segmentation of 3D Medical Images. Y. Bizais et al. (eds.): Information Processing in Medical Imaging, Kluwer Academic Publishers 1995

[27] J. A. Marchant: Adding Grey Level Information to Point Distribution Models Using Finite Elements. Proc. British Machine Vision Conference, BMVA Press 1993

[28] T. McInerney, D. Terzopoulos: Medical Image Segmentation Using Topologically Adaptable Snakes. N. Ayache (ed.): Computer Vision, Virtual Reality and Robotics in Medicine, First Int. Conf., CVRMed'95, Nice

[29] P. Mrázek: Genetic Algorithms and Image Search. N. M. Thalmann, V. Skala (ed.): WSCG'96, Fourth Int. Conference in Central Europe on Computer Graphics and Visualisation, Plzen

[30] P. Mrázek, P. Felkel, L. Sýkora: Model-Based Segmentation of Medical Images. 11. Spring Conference on Computer Graphics, SCCG'95, Bratislava, Proceedings

[31] Nastar, Ayache: Non-Rigid Motion Analysis in Medical Images: a Physically Based Approach. In: IPMI'93 (Reference [3])

[32] Parvin B., Johnston W.: Pinta: A System for Visualising Anatomical Structures of the Brain from MR Imaging. SPIE Vol. 2167 Image Processing (1994)

[33] W. H. Press et al.: Numerical Recipes in C: the art of scientific computing. Cambridge University Press 1988, 1992.

[34] S. P. Raya: Low-Level Segmentation of 3-D Magnetic Resonance Brain Images - A Rule-Based System. IEEE Trans. on Medical Imaging, Vol. 9, No. 3, Sept. 1990

[35] G. P. Robinson, Colchester, Griffin: Model-Based Recognition of Anatomical Objects from Medical Images. In: IPMI'93 (Reference [3])

[36] S. R. Sandor, R. M. Leahy: Towards Automated Labelling of the Cerebral Cortex Using a Deformable Atlas. Y. Bizais et al. (eds.): Information Processing in Medical Imaging, Kluwer Academic Publishers 1995

[37] Schubert, Hoehne, Riemer, Schiemann, Tiede: Spatial Knowledge Representation for Visualisation of Human Anatomy and Function. In: IPMI'93 (Reference [3])

[38] D. Y. Suh, Eisner, Mersereau, Pettigrew: Knowledge-Based System for Boundary Detection of Four-Dimensional Cardiac Magnetic Resonance Image Sequences. IEEE Trans. on Medical Imaging, Vol. 12, No. 1, March 1993

[39] G. Székely, A. Kelemen, et al.: Segmentation of 3D Objects from MRI Volume Data Using Constrained Elastic Deformations of Flexible Fourier Surface Models. N. Ayache (ed.): Computer Vision, Virtual Reality and Robotics in Medicine, First Int. Conf., CVRMed'95, Nice

[40] J. Tailarach and P. Tournoux: Co-planar stereotaxic atlas of the human brain: 3-Dimensional proportional system: an approach to cerebral imaging. Georg Thieme Verlag, Stuttgart, New York, 1988.

[41] D. Terzopulos, D. Metaxas: Dynamic 3D Models with Local and Global Deformations: Deformable Superquadrics. IEEE Trans. PAMI, Vol.13, No.7, July 1991, pp.703-714

[42] S. Umeyama: Least-Squares Estimation of Transformation Parameters Between Two Point Patterns. IEEE Trans. PAMI, Vol. 13, No. 4, April 1991

[43] Vandermeulen, Verbeeck, Berben, Suetens, Marchal: Continuous Voxel Classification by Stochastic Relaxation: Theory and Application to MR Imaging and MR Angiography. In: IPMI'93 (Reference [3])

[44] B. C. Vemuri, A. Radisavljevic, C. M. Leonard: Mulri-resolution Stochastic 3-D Shape Models for Image Segmentation. In: IPMI'93 (Reference [3])

[45] A. Watt, M. Watt: Advanced Animation and Rendering Techniques - Theory and Practice. Addison-Wesley 1992.

[46] Worring, Smeulders, Staib, Duncan: Parametrized feasible boundaries in gradient vector fields. In: IPMI'93 (Reference [3])

[47] J. Žára a kol.: Počítačévá grafika - principy a algoritmy. Grada 1992.

[48] Žižka J.: Zobrazování magnetickou rezonancí - principy a aplikace. Čes. Radiol., 48, 1994, p. 260-275

[49] Smullyan R. M.: Jak se jmenuje tahle knížka (approx. What's the name of this book). Mladá fronta 1986

[50] Johnson, Wichern: Multivariate Statistical Analysis. Chapman&Hall 1988.

[51] B. Castiglione: Etiquette for Renaissance Gentlemen. Penguin Books 1967, 1995.