Mathematical Optimization in Graphics and Vision

Luiz Velho

Paulo Cezar Pinto Carvalho

Instituto de Matemática Pura e Aplicada, IMPA
Rio de Janeiro, Brazil

Course Notes – SIGGRAPH 2002

Course Description

Mathematical optimization has a fundamental importance for the solution of many problems in computer graphics and vision. This fact is apparent from a quick look at the SIGGRAPH proceedings, where a significant percentage of the papers employ in one way or another mathematical optimization techniques.

This course will provide a conceptual analysis of the problems in computer graphics and discuss the mathematical models used to solve them. The goal is to develop an understanding of the importance of optimization techniques in graphics and vision.

The course will also give an overview of combinatorial, continuous and variational optimization methods, focusing on graphical applications.

The prerequisites for the course are: (i) background on linear algebra and calculus of one and several variables; (ii) computational experience on algorithms and programming; (iii) knowledge of geometric modeling, animation, image processing, image analysis and visualization.

These notes originated from a set of notes in Portuguese written for a course on this topic at the Brazilian Mathematical Colloquium in July of 1999 and at Brazilian Congress of Applied Mathematics in October of 2000. We wish to thank Jonas Gomes and Luiz Henrique de Figueiredo who collaborated with us to produce the portuguese notes.

Syllabus

This tutorial is divided into two parts. The first part will give an overview of computer graphics, focusing on problems and describing the mathematical models used to solve them. This is a short introduction to motivate the study of optimization techniques. The subject will be taught in such a way to make it clear the need to use optimization techniques in the solution of a large family of problems.

The second part will introduce the subject of optimization and provide a classification of the optimization techniques into different categories: continuous, variational and combinatorial methods. Several examples in computer graphics will be given to illustrate each category of problems.

Part 1: Optimization problems in graphics

Duration: 50 minutes
Speaker: Luiz Velho

Part 2: Overview of optimization techniques

Duration: 50 minutes
Speaker: Paulo Cezar Pinto Carvalho

Course Speakers Biographies

Luiz Velho

Luiz Velho is an Associate Researcher at IMPA -Instituto de Matematica Pura e Aplicada.He received a BE in Industrial Design from ESDI - Universidade do Rio de Janeiro in 1979, a MS in Computer Graphics from the Massachusetts Institute of Technology, Media Laboratory in 1985, and a Ph.D.in Computer Science in 1994 from the University of Toronto. His experience in computer graphics spans the fields of modeling, rendering, imaging and animation.

During 1982 he was a visiting researcher at the National Film Board of Canada. From 1985 to 1987 he was a Systems Engineer at the Fantastic Animation Machine in New York, where he developed the companyś 3D visualization system. From 1987 to 1991 he was a Principal Engineer at Globo TV Network in Brazil, where he created special effects and visual simulation systems. In 1994 he was a visiting professor at the Courant Institute of Mathematical Sciences, New York University.

He is the author of several books in graphics, and has published several papers in this area. He has been a speaker in previous SIGGRAPH courses: Modeling in Graphics in SIGGRAPH 93, Warping and Morphing of Graphical Objects in SIGGRAPH 94 and SIGGRAPH 97. He was the co-organizer of the course From Fourier to Wavelets in SIGGRAPH 98 and 99.

Paulo Cezar Pinto Carvalho

Paulo Carvalho is an Associate Researcher at IMPA, Instituto de Matematica Pura e Aplicada. He received a bachelors degree in Civil Engineering from IME (Rio de Janeiro) in 1975, a MS in Statistics from IMPA in 1980, and a PhD in Operations Research from Cornell University in 1984.

His research interests include Computational Geometry, several fields in Computer Graphics, especially modeling and visualization, and Computer Vision, having published several papers on these subjects. He is also very active in promoting the improvement of Mathematics education at all levels and has written several books aimed at Mathematics teachers. He has been a researcher at IMPA since 1985. In 1987 and 1988 he was a Visiting Professor at Cornell University, having taught several undergraduate and graduate level courses.

Since 1994 he has been a consultant to TeCGraf, a group that specializes in providing assistance to the industry in developing advanced graphics applications. He has given courses on Computational Geometry, Modeling and Optimization in Computer Graphics in conferences such as the Brazilian Mathematics Colloquium(1991, 1999) and the Brazilian Symposium On Computer Graphics and Image Processing, SIBGRAPI (1991, 1992).

Contents

List of Figures

Chapter 1

Computer Graphics

The usual definition for computer graphics is the following: *a set of techniques to transform data into images using a graphics device*. The attempt to define an area is a difficult task. In that respect, perhaps the best way to understand an area is through a deep knowledge of its problems and the methods to solve them. From this point of view, the definition above has the virtue of emphasizing a fundamental problem of computer graphics, that is: *the transformation of data into images*. (Figure 1.1).

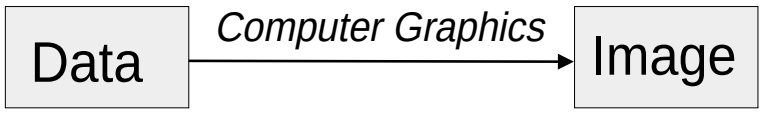


Figure 1.1: Computer graphics: conversion of data into images.

In applied mathematics, the solution of problems is directly related with the mathematical models used to understand the problem. In this case, the dividing line between solved and open problems is more subtle than in the case of pure mathematics, where different solutions to the same problem, in general, do not constitute great innovations from the scientific point of view. On the other hand, in applied mathematics, different solutions to the same problem arise as consequence of the use of new models, and usually bring a significant advance in terms of applications.

These notes discuss the solution of various problems in computer graphics and vision using mathematical optimization techniques. The idea of this text is to serve as a two-way channel: on one hand, stimulate the computer graphics community to study optimization methods; and on the other hand, call attention of the optimization community for the extremely interesting problems in graphics and vision.

1.1 Related Areas

Since its origin, Computer Graphics studies methods that allow the visualization of information using a computer. Because, in practice, there is no limitation to the origin or nature of the data, computer graphics is used today by researchers and users from many different areas of human activity.

The basic elements of computer graphics are: “data” and “images”. There are four related areas that deal with these elements, as illustrated in Figure 1.2.

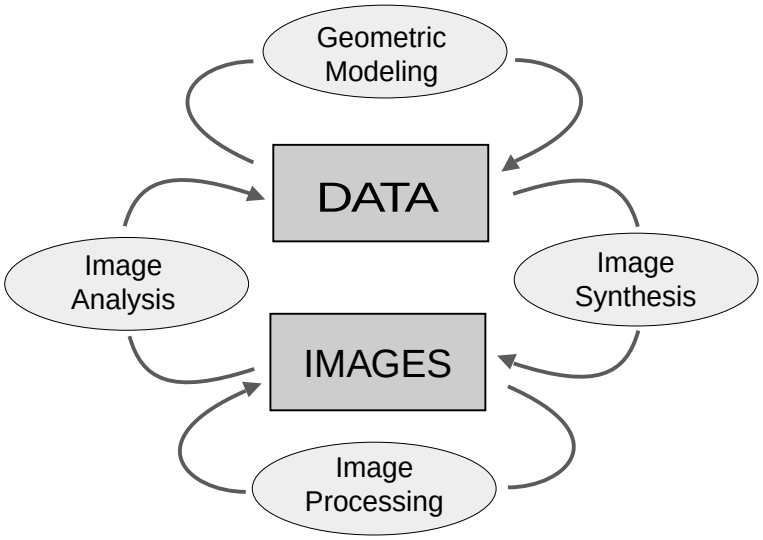


Figure 1.2: Computer graphics: related areas.

Geometric Modeling, deals with the problem of describing and structuring geometric data in the computer.

Visualization, interprets the data created by geometric modeling to generate an image that can be viewed using a graphical output device.

Image Processing, transforms input images into output images. Examples of these transformations are: enhancement, colorization, etc.

Computer Vision, extracts from an input image various types of information (geometric, topological, physical, etc) about the objects depicted in the image.

In the literature, it is common to consider computer graphics as the area above called visualization. We believe it is more convenient to consider computer graphics as the "mother area" which encompasses these four sub-areas above.

This is justified, because these related areas work with the same objects (models and images). Consequently there is a strong trend of integration. Moreover, the solution of certain problems require the use of methods from all these areas under a unified framework.

As an example, we can mention the case of a GIS (Geographical Information System) application, where a satellite image is used to obtain the elevation data of a terrain model and a three-dimensional reconstruction is visualized with texture mapping from different view points.

This combination of techniques from related areas is exactly where it lies a great potential for advance — the combined result is more than the sum of parts.

The trend is so vigorous that new research areas have been created. This was the case of *image-based modeling and rendering*, a recent area that combines techniques from computer graphics, geometric modeling, image processing and vision.

Furthermore, in some application domains such as GIS and Medical Imaging, it is natural to use techniques from these related areas, to the point that there is not a separation between them.

We intend to show in these notes how mathematical optimization methods can be used in a unified framework involving these related areas.

1.2 Graphics Objects

We would like to define computer graphics as the area that deals with *description, analysis and processing of graphics objects*. This definition only makes sense if we are able to establish in a precise manner the notion of a graphics object.

From the mathematical point of view, a *graphics object* is a subset $S \subset \mathbb{R}^m$ together with a function $f\colon S \subset \mathbb{R}^m \to \mathbb{R}^n$. The set S is called *geometric support*, and f is called *attribute function* of the graphics object. The dimension of the geometric support S is called *dimension* of the graphics object. This concept of graphics object was introduced in the literature by (Gomes *et al.* , 1996).

Let's see some concrete examples to clarify these notions.

Example 1. *(*Subsets of Space) Any subset of the euclidean space $\mathbb{R}^m$ is a graphics object. Indeed, given $S \subset \mathbb{R}^m$, we define immediately an attribute function

$$f(p) = \begin{cases} 1 & \text{if } p \in S, \\ 0 & \text{if } p \notin S. \end{cases}$$

It is clear that $p \in S$ if, and only if, $f(p) = 1$. In general, the values of $f(p) = 1$ are associated with a color, called *object color*. The attribute function in this case simply characterizes the points of the set S, and for this reason, it is called *characteristic function* of the graphics object.

The characteristic function completely defines the geometric support of the graphics object, that is, *if p is a point of the space $\mathbb{R}^m$, then $p \in S$ if, and only if, $f(p) = 1$.*

The two problems below are therefore, equivalent:

1. devise an algorithm to compute $f(p)$ at any point $p \in \mathbb{R}^m$;

2. determine if a point $p \in \mathbb{R}^n$ belongs to the geometric support S of the graphics object.

The second problem is called *point-membership classification problem*. A significant part of the problems in the study of graphics objects reduce to a solution of point-membership classification. Therefore, the existence of robust and efficient algorithms to solve this problem is of great importance.

Example 2. *(*Image) An image is a function $f\colon U \subset \mathbb{R}^2 \to \mathbb{R}^n$, where $\mathbb{R}^n$ is the representation of a color space. In this way, we see that an image is a graphics object whose geometric support is the subset U of the plane (in general, a rectangle), and the attribute function associates a color to each point of the plane.

Example 3. *(*Circle and vector field) Consider the unit circle S^1 centered at the origin, whose equation is given by

$$x^2 + y^2 = 1.$$

A mapping of the plane $N\colon \mathbb{R}^2 \to \mathbb{R}^2$, given by $N(x, y) = (x, y)$, defines a field of unit vectors normal to S^1. The map $T\colon \mathbb{R}^2 \to \mathbb{R}^2$, given by $T(x, y) = (y, -x)$, defines a field of vectors tangent to the circle. (Figure 1.3).

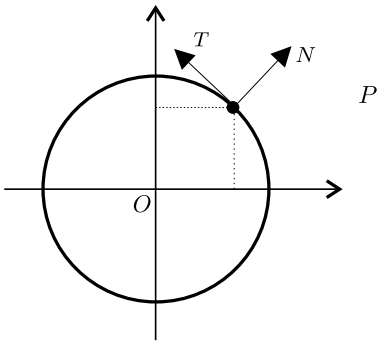


Figure 1.3: Circle with normal and tangent vector fields.

The circle is a one-dimensional graphics object of the plane, and the two vector fields are attributes of the circle (they can represent, for example, physical attributes such as tangential and radial acceleration). The attribute function is given by $f\colon S^1 \to \mathbb{R}^4 = \mathbb{R}^2 \oplus \mathbb{R}^2$, $f(p) = (T(p), N(p))$.

1.3 Description, Representation and Reconstruction

There are two basic methods to describe graphics objects: parametric and implicit. More details about these methods can be seen in (Gomes & Velho, 1998).

The representation of graphics objects constitute a separate chapter in computer graphics. Because discretization is the essence of computational methods, representation techniques assume great importance in the area.

In many situations, it is convenient to work with continuous models of graphics objects. The reader may find strange to talk about continuous object in the context of computer applications. Let's make this clear: we say that an object is continuous when we are able to obtain the coordinates of any point of the object and compute the value of its attribute function at this point. The operation to recover the continuous object from its discrete representation is called *reconstruction* (see Figure 1.4).

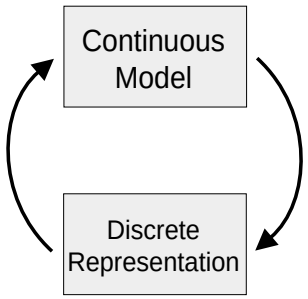


Figure 1.4: Representation and reconstruction.

We give below four reasons to highlight the importance of reconstruction:

1. When we need to obtain an alternative representation of an object, we can reconstruct it and then build a new representation of the object.

2. The reconstruction is useful when we need to work in the continuous domain to minimize numeric errors.

3. The reconstruction is fundamental in the visualization process.

4. In general, the user specifies a graphics object directly on some representation, since an user interface allows only the input of a finite set of parameters. The object must be reconstructed from this specification.

The reconstruction operation is totally dependent of the representation and, in general, it is difficult to be computed. When a given representation method allows a reconstruction that recovers the original object, we call it an *exact representation*.

Exact representations are rare. In general, reconstruction methods can only provide approximations of the original objects. In the next chapters we will investigate how optimization techniques allow us to obtain good reconstructions of graphics objects.

Chapter 2

Optimization Problems in Graphics

In this chapter we will show the importance of optimization methods in computer graphics. The exposition will be based on an analysis of the main problems of the area.

2.1 Solving Problems

The various problems in computer graphics can be formulated using the notion of operators between spaces of graphics objects. These problems can be classified in two categories: *direct* and *inverse* problems.

Direct problems. Given two spaces $\mathcal{O}_1$ and $\mathcal{O}_2$ of graphic objects, an operator $T\colon \mathcal{O}_1 \to \mathcal{O}_2$, and a graphic object $x \in \mathcal{O}_1$. Problem: determine the object $y = T(x) \in \mathcal{O}_2$.

Inverse problems. There are two types of inverse problems:

1. Given two spaces $\mathcal{O}_1$ and $\mathcal{O}_2$ of graphics objects, an operator $T\colon \mathcal{O}_1 \to \mathcal{O}_2$, and an element $y \in \mathcal{O}_2$, determine an object $x \in \mathcal{O}_1$ such that $T(x) = y$.

2. Given elements $x \in \mathcal{O}_1$ e $y \in \mathcal{O}_2$, determine an operator T such that $T(x) = y$.

Hadamard[1] has established the concept of *well-posed* problem as a problem in which the following conditions are satisfied:

1. There is a solution;

2. The solution is unique;

3. The solution depends continuously on the initial conditions.

When at least one of the above conditions is not satisfied, we say that the problem is *ill-posed*.

The concept of ill-posed problem needs to be further investigated very carefully. The continuity condition is important because it guarantees that small variations of the initial condition will not alter too much the solution. Recall that, in practical problems, small variations in the initial conditions are quite common (and sometimes, inevitable) due to measurement imprecision or numerical errors. The non-uniqueness of solutions, on the other hand, should not be necessarily an obstacle. An example will make this point clear.

Example 4. The equation $x^2 + y^2 - 1 = 0$ admits an infinity of solutions, being ill-posed in the sense of Hadamard. However, in order to obtain a sampling of the circle, approximating it by an inscribed polygon, we need to determine a subset of these solutions. Figure 2.1 shows the use of seven solutions of the equation that allows us to represent the circle by an heptagon.

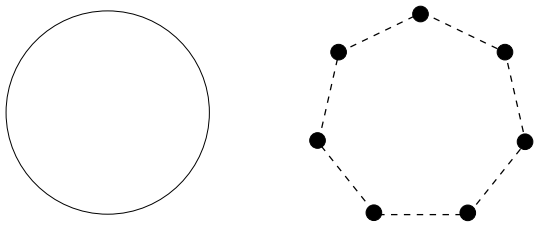


Figure 2.1: Polygonal representation of a circle.

[1] Jacques Hadamard (1865-1963), French mathematician.

Observe that the infinite number of solutions of sampling problem in the example above are related with an inverse problem. We can say that, in general, inverse problems are ill-posed in the sense of Hadamard, particularly because of the non-uniqueness of the solution,

The use of the term "ill-posed" to designate the class of problems that do not satisfy one of the three Hadamard conditions may induce the reader to conclude that ill-posed problems cannot be solved, but this is far from the truth. By interpreting ill-posed problems as a category of intractable problems, the reader would be discouraged to study computer graphics, an area in which there is a large number of ill-posed problems, as we will see in this chapter.

In fact, ill-posed problems constitute a fertile ground for the use of optimization methods. When a problem presents an infinite number of solutions and we wish to have uniqueness, optimization methods make possible to reformulate the problem, so that a unique solution can be obtained. The general principle is to define an objective function such that an "optimal solution" can be chosen among the set of possible solutions. In this way, when we do not have only one solution to a problem we can have the "best" solution, (which is unique).

It is interesting to note that, sometimes, well-posed problems may not present a solution due to the presence of noise or numerical errors. The example below shows one such case.

Example 5. Consider the solution of the linear system

$$\begin{aligned} x + 3y &= a \\ 2x + 6y &= 2a, \end{aligned}$$

where $a \in \mathbb{R}$. In other words, we want to solve the inverse problem $TX = A$, where T is the linear transformation given by the matrix

$$\begin{pmatrix} 1 & 3 \\ 2 & 6 \end{pmatrix},$$

$X = (x, y)$ and $A = (a, 2a)$. It is easy to see that the image of $\mathbb{R}^2$ by T is the line r given by the equation $2x - y = 0$. Because the point A belongs to the line r, the problem has a solution (in fact an infinite number of solutions).

However, a small perturbation in the value of a can move the point A out of the line r, and the problem fails to have a solution.

To avoid this situation, a more appropriate way to pose the problem employs a formulation based on optimization: *determine the element $P \in \mathbb{R}^2$ such that the distance from $T(P)$ to the point A is minimal*. This form of the problem always has a solution. Moreover, this solution will be close to the point A. Geometrically, the solution is the point P, such that its image $T(P)$ is the closest point of the line r to the point A (see Figure 2.2).

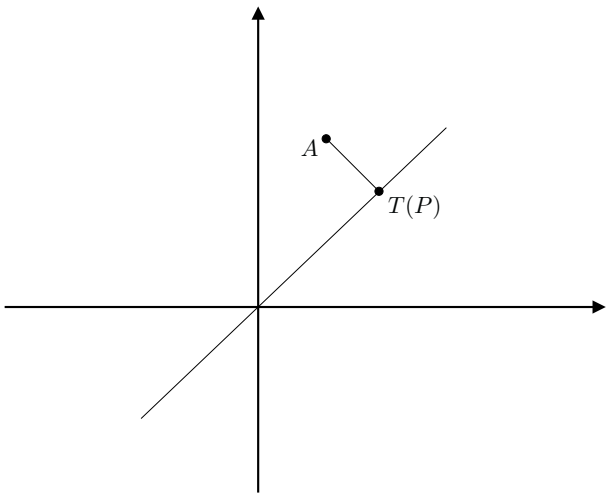


Figure 2.2: Well posed formulation of a problem.

2.2 Representation and reconstruction

We saw in Chapter 1 that one of the important problems in computer graphics is the representation and reconstruction of graphics objects. From the mathematical point of view, we can formulate this problem in the following way: Given an space of graphics objects $\mathcal{O}_1$ and an space of discrete graphics objects $\mathcal{O}_2$, an operator $R\colon \mathcal{O}_1 \to \mathcal{O}_2$ is called a *representation operator*. This operator associates, to each graphics object $x \in \mathcal{O}_1$, its discrete representation $R(x) \in \mathcal{O}_2$.

Given an object $y \in \mathcal{O}_2$, a *reconstruction* of y is a graphics object $x \in \mathcal{O}_1$ such that $R(x) = y$. The reconstruction of y is indicated by $R^+(y)$. When R is

an invertible operator, we have that $R^{+}(y) = R^{-1}(y)$. Note, however, that invertibility is not a necessary condition for reconstruction. In fact, it suffices that the operator R possesses a left inverse, $R^{+}(R(x)) = x$, to make reconstruction possible.

Note that the representation problem is a direct problem, while the reconstruction problem is an inverse problem. A representation that allows more than one possibility of reconstruction is called an *ambiguous representation* [2]. In this case, the reconstruction problem is ill-posed (it has more than one solution).

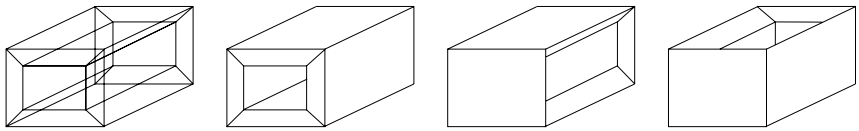
Although the representation problem constitutes a direct problem, the computation of the representation operator could lead to an inverse problem. An example of such a situation can be seen in the point sampling representation of an implicit shape, $F^{-1}(0)$, where $F\colon \mathbb{R}^3 \rightarrow \mathbb{R}$. In this case, the representation consists in obtaining solutions of the equation $F(x, y, z) = 0$, which constitutes an inverse problem.

2.2.1 Semantics and Reconstruction

The reconstruction of a graphics object determines its semantics. Therefore, it is of great importance in the various processes of computer graphics. We can listen to a sound that is reconstructed by a loudspeaker; and we see an image that is reconstructed on the screen of a graphics display or printed on paper, etc.

In this way, reconstruction methods play a fundamental role in visualization. It is desirable to avoid ambiguous representations, which can make the reconstruction problem non-unique. A geometric example of an ambiguous representation is shown in Figure 2.3, where the object in image (a) is represented by a wireframe model. In images (b), (c) and (d), we can see three possible reconstructions of this object.

[2] It would be more correct to adopt the term “ambiguous reconstruction”, but the term “ambiguous representation” is already widely adopted in computer graphics.



(a) (b) (c) (d)

Figure 2.3: Ambiguous representation.

2.3 Geometric Modeling

Large part of the problems in modeling are related to the description, representation, and reconstruction of geometric models. There are essentially three methods to describe a geometric shape:

1. implicit description;
2. parametric description;
3. piecewise description (implicit or parametric);

The description is usually given in functional form, either deterministic or probabilistic.

A model is called *procedural* when its describing function is an algorithm over some virtual machine.

In general, the geometry of a model is specified by the user through a finite number of parameters; based on this specification the model is reconstructed.

Some common cases are:

- reconstruction of a curve or surface from a set o points. In this case the reconstruction employs some interpolation method for scattered data;
- reconstruction of surfaces based on a finite set of curves.

It is clear that the two problems above lack uniqueness of solution. Additional conditions are required for uniqueness. These conditions can be obtained by algebraic methods or using optimization. In the first case, we may impose that the curve or surface belong to some specific class, for example, defined by a

polynomial of degree 3. In the second case, we may look for a curve or surface that minimizes some given objective function.

The methods to create models using optimization are known in the literature by the name of *variational modeling*. Let's investigate a concrete example for the case of planar curves.

2.3.1 Variational Modeling of Curves

The basic problem of describing curves is related to the creation of geometric objects in the context of specific applications. A generic method consists in determining a curve that passes through a finite set of points $p_1, \ldots, p_n$ and at the same time satisfy some application-dependent criteria. In general, besides obtaining a curve with a given shape, many other conditions that measure the 'quality" of the curve can be imposed. Among those conditions we could mention:

- Continuity;
- Tangent line at control points;
- Continuous curvature, etc.

Each condition imposes some restriction on the curve to be constructed. These restrictions can be of analytical, geometric or topological nature. A convenient way to obtain curves that satisfy some set of conditions is to pose the problem in the context of optimization: define an energy function E such that the curves which are minimizers of this functional automatically satisfy the desired criteria.

D. Bernoulli[3] was the first mathematician to propose a functional to measure the energy associated with the tension of a curve. This energy is called *tension energy* and is proportional to the square of the curvature k of the curve:

$$\mathrm{E}_{\mathrm{tension}}(\alpha) = \int_{\alpha} k^2 ds, \tag{2.1}$$

[3]Daniel Bernoulli (1700-1782), Swiss mathematician known by its work in hydrodynamics and mechanics.

where s is the arc length of the curve α. Note that the tension energy of a straight line is zero because its curvature is identically zero. On the other hand, if the tension energy of a curve is high, then it "bends" significantly. Curves that minimize the tension energy are called *non-linear splines*.

The geometric restrictions that are imposed can be of punctual or directional nature. An example of punctual restriction is to force the curve to pass through a finite set of points. An example of directional restriction is to constraint the tangent vector of the curve to have a certain direction, previously specified.

The geometric restrictions are in general modeled by an energy functional associated with external forces that act on the curve. The tension energy discussed earlier is an internal energy to the curve. In this way, we have a unification of the optimization problem looking for the minima of an energy functional E that has an internal component and an external component:

$$\mathrm{E}_{\mathrm{total}}(\alpha) = \mathrm{E}_{\mathrm{int}}(\alpha) + \mathrm{E}_{\mathrm{ext}}(\alpha).$$

A simplification adopted very often consists in decomposing the internal energy into a *bending* and an *stretch* components. In this way, the internal energy is given by the linear combination

$$\mathrm{E}_{\mathrm{int}}(\alpha) = \mu\,\mathrm{E}_{\mathrm{bending}} + (1-\mu)\,\mathrm{E}_{\mathrm{stretch}},$$

$\mu \in [0,1]$. The bending energy is given by equation (2.1), and the stretch energy i given by

$$\mathrm{E}_{\mathrm{stretch}}(\alpha) = \int_{\alpha} ||\alpha'(t)||\, dt.$$

Therefore, the internal energy controls the energy accumulated by the curve when it is bended or stretched. A minimization of this functional results in a curve that is as short and straight as possible. If the curve is constrained to pass through two points, the solution of the minimization problem is certainly a line segment. The external energy results in non-trivial solutions to the problem.

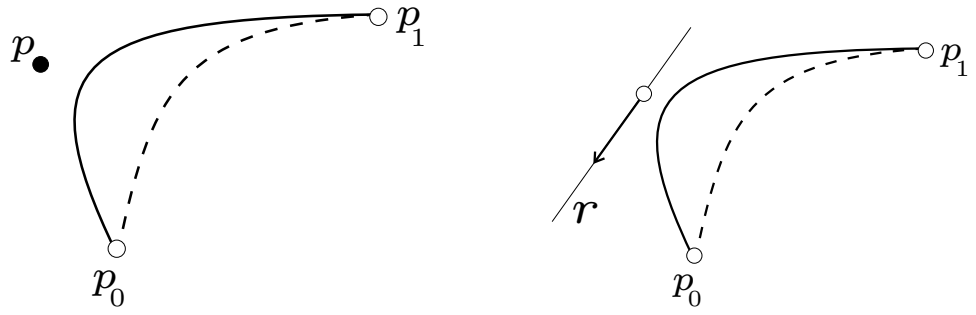
Intuitively, the external energy deforms the curve in many ways, bending and stretching it. In general, the user specifies various components of external energy, which are linearly combined in order to obtain the desired deformation. This energy is defined in the form of attractors or repulsors, which can be of two

kinds: *positional* or *directional*. We will give below two examples to clarify these concepts.

Example 6. ***(*Punctual attractors.)*** An example of a positional attractor is the punctual attractor, whose energy functional is given by

$$\mathrm{E}_{\mathrm{punctual}}(\alpha) = d(\alpha, p)^2 = min_t||\alpha(t) - p||^2.$$

Figure 2.4(a) shows an example of a curve that passes through points p_0 and p_1 while being deformed by attraction to point p.



(a) (b)

Figure 2.4: Attractors: punctual (a) and directional (b).

Example 7. ***(*Directional attractor.)*** An example of a directional attractor is based on the specification of a line $r(t) = p + tv$. The curve starts at point p_0 and ends at point p_1. The external energy forces the curve to approach point p and the tangent to the curve at points near p will line up with vector v. (see Figure 2.4(b)). This condition can be obtained by an energy functional given by the equation

$$\mathrm{E}_{\mathrm{dir}}(\alpha) = \min_t ||\alpha'(t) \wedge v||^2.$$

2.4 Visualization and Computer Vision

The visualization process consists in the generation of images from geometric models. Formally, we define a *rendering operator* which associates to each point in the ambient space to a corresponding point in the image with its respective color.

However, the solution of this direct problem is not simple, because the rendering operator depends on many factors: i.e. light sources, camera transformation, geometry and material properties of the objects of the scene, and ultimately of the propagation of radiant energy through the scene reaching the image plane. Depending on the type of illumination model adopted, the rendering computation can be a direct or inverse problem.

Computer vision is clearly an inverse problem. In this area we are looking for physical, geometrical or topological information about the objects of a scene which is depicted in an image. In a certain way, the image is a representation of the scene, and the vision problem consists in the reconstruction of the three-dimensional scene from its two-dimensional representation. In the visualization process that generates an image there is a great loss of information implied by the projection that changes the dimensionality of the representation. The image, therefore, is an extremely ambiguous representation.

Even the reconstruction of the scene by our sophisticated visual system may present ambiguities. Two classic examples of such semantic ambiguity are shown in Figure 2.5. In (a) we can see an image which can be interpreted either as the face of an elderly woman or of a young lady (the nose of the old woman is the chin of the lady). In (b) we can interpret the figure either as the silhouette of two faces over a white background or as a white glass over a black background. Note that the ambiguity of the second image is related with the problem of distinguishing between foreground and background in the reconstruction.

(a) (b)

Figure 2.5: Ambiguous reconstructions.

The complexity of reconstruction in computer vision originates different versions of the reconstruction problem. Two important cases of the problem are:

- **Shape from shading:** Obtain the geometry of the objects based on information about the radiance of the pixels in the image.
- **Shape from texture:** Obtain information about the geometry of the scene based on a segmentation and analysis of the textures in the image.

2.4.1 The Virtual Camera

One of the important elements of the visualization process is the virtual camera. The simplest model of a virtual camera is the "pinhole" camera, that has seven degrees of freedom representing the parameters:

- position (3 degrees of freedom);
- orientation (3 degrees of freedom);
- focal distance (1 degree of freedom).

These parameters determine a projective transformation T which associates to each point $p \in \mathbb{R}^3$ a point in the image plane (for more details, see (Gomes

& Velho, 1998)). In other words, given a point $p \in \mathbb{R}^3$, the virtual camera transformation generates the point $T(p) = p'$ in the image.

We have two inverse problems related in a natural way with the virtual camera:

1. Given the point p' in the image, determine the point p of the scene, assuming that the camera parameters are known.

2. Given points p and p', determine the camera transformation T such that $T(p) = p'$.

In the first problem, we could have many points in the scene which correspond to the point p' in the image (i.e. 3D points that are projected in the same 2D image point). In the second case, it is clear that the knowledge of just one 3D point and its 2D projection is not sufficient to determine the seven camera parameters.

One important problem is the application of (2) above to the case of images produced by real cameras. In other words, given an image acquired by some input device (video camera or photographic camera), determine the camera parameters (position, orientation, focal length, etc.). This problem is known as *camera calibration*.

Camera calibration has a large number of applications. Among them we could mention:

- In virtual reality applications, it is often necessary to combine virtual images with real ones. In this case, we need to synchronize the parameters of the virtual camera with those of the real camera in order to obtain a correct alignment of the elements in the image composition.

- Some programs that are used in broadcast transmission of soccer games need to know the camera parameters in order to create a 3D reconstruction of the players in the field. More information about this type of application can be obtained in the website *juiz virtual*, `http://www.visgraf.impa.br/juizvirtual/`.

Camera Specification

An important problem in graphics is the camera specification. In the direct specification, the user provides the seven camera parameter that are required to define the transformation T. Then, we have the direct problem of computing image points $p' = T(p)$.

A related inverse problem arises in the study of user interfaces for camera specification. In general, as we have seen, the user specifies the transformation T though the seven camera parameters, leading to an easy to solve direct problem. However, this interface is not appropriate when the user wants to obtain specific views of the scene (for example, keeping the focus of attention on a certain object). A suitable interface should allow the user to frame an object directly on the image. Such a technique is called *inverse camera specification*.

In the inverse specification, the seven degrees of freedom of the camera are specified *indirectly* by the user. The user wishes to obtain a certain framing of the scene and the specification must produce the desired result.

The idea of the inverse specification is to allow the definition of camera parameters inspired on the "camera man paradigm": the user (in the role of a film director) describes the desired view directly on the image, and the system adjusts the camera parameters to obtain that view.

The inverse camera specification facilitates the view description by the user at the cost of having to solve an ill-posed mathematical problem. Let's see how it would be the solution of such a problem through a concrete example.

Example 8. ***(*Framing a point in the image.)** Consider the situation illustrated in Figure 2.6. Point P in space is fixed and the observer is located at point O, this point is projected in point A on the screen. The user requests a new view, subject to the constraint that point A is at the center of the image. Therefore, we need to obtain camera parameters, such that point P will now be projected in point B which is located at the center of the screen

Figure 2.6 shows a new position O', and a new orientation of the camera that solves the problem. Observe that this camera setting is not the unique solution to the problem.

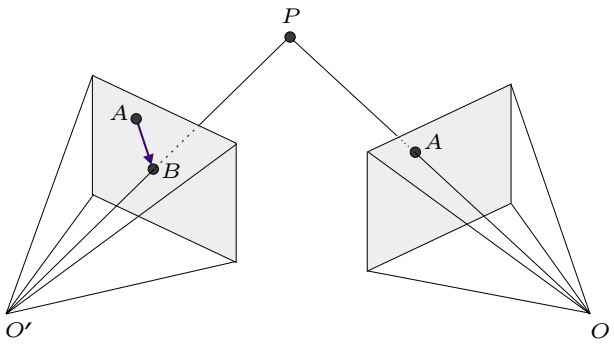


Figure 2.6: Inverse specification to frame a point in the image (Gomes & Velho, 1998).

From the mathematical point of view we have a transformation $T\colon \mathbb{R}^7 \to \mathbb{R}^2$, where $y = T(x)$ is the projection of the parametrization space of the camera on the euclidean plane. The solution to our problem is given by the inverse $T^{-1}(B)$. However, the transformation T is not linear and, in general, it does not have an inverse (why?). Therefore, the solution has to be computed using optimization techniques. That is, we need to look for the "best" solution in some sense.

2.5 Image Processing and Analysis

In this section we will discuss some problems related with the analysis and processing of images.

2.5.1 Image Processing

Image processing studies different operations with images. These operations are characterized by operators in spaces of images.

An important operation consists in the warping of the domain U of an image $f\colon U \subset \mathbb{R}^2 \to \mathbb{R}$. More specifically, given an image $f\colon U \to \mathbb{R}$, a warping of U is a diffeomorphism $g\colon V \subset \mathbb{R}^2 \to U$. This diffeomorphism defines a new image given by $h\colon V \to \mathbb{R}$, $h = f \circ g^{-1}$. An example of such an operation is illustrated in Figure 2.7.

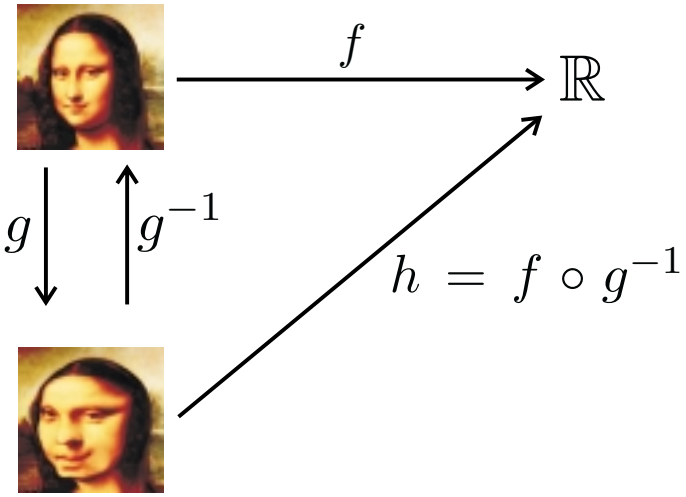


Figure 2.7: Image deformation.

Image warping has many applications. Among them, we could mention: image morphing, registration and correction.

Image Morphing. It deforms an image into another in a continuous way (see Figure 2.8);

Figure 2.8: Image metamorphosis.

Registration of medical images. It is very common in diagnosis the comparison of different medical images. It is also common to combine images of the same part of the body obtained from different sources. In all these cases, the images need to be perfectly aligned.

Correcting image distortions. Many times, the process of capturing or generating an image introduces distortions. In such cases, it is necessary to correct the distortions with a warping. We employ a standard pattern to measure distortions and based on those measures we correct other images produced by the process. Figure 2.9 shows an example of a standard pattern and its deformation due to some image capture process.

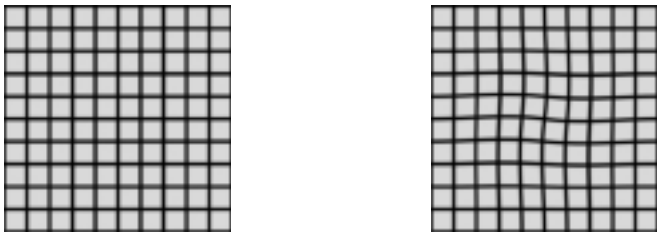


Figure 2.9: Correcting distortions.

One of the main problems in the area of warping and morphing is the specification of the warping operator. There are many techniques for this purpose. A simple and very popular method is the punctual specification, illustrated in Figure 2.10: In order to deform the bird into the dog, we select some points in the outline curve of the bird and their corresponding points in the outline of the dog. Through the knowledge of the deformation acting on this set of points it is possible to reconstruct the desired deformation over the whole space.

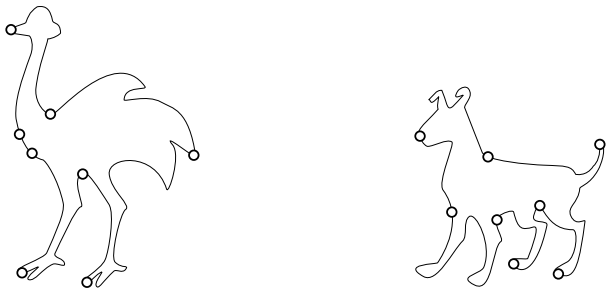


Figure 2.10: Punctual specification.

With punctual specification, the problem of finding a deformation reduces to the problem of determining a warping $g\colon \mathbb{R}^2 \to \mathbb{R}^2$ such that

$$g(p_i) = q_i, \quad i = 1, \ldots, n,$$

where p_i and q_i are points of $\mathbb{R}^2$. It is clear that this problem does not admit a unique solution, in general.

Suppose that $g(x, y) = (g_1(x, y), g_2(x, y))$, where $g_1, g_2\colon U \to \mathbb{R}$ are the components of g. In this case, the equations $g(p_i) = q_i$ can be written in the form

$$\begin{aligned} g_1(p_x^i, p_y^i) &= q_x^i; \\ g_2(p_x^i, p_y^i) &= q_y^i, \end{aligned}$$

with $i = 1, \ldots, n$. That is, the problem amounts to determining two surfaces

$$(x, y, g_1(x, y)), \quad (x, y, g_2(x, y)),$$

based on the knowledge of a finite number of points on each surface. Note that we encounter this same problem in the area of modeling.

The way that the deformation problem was posed using point specification is too rigid: If we restrict the class of warpings that are allowed, the problem may not have a solution. In addition, the above formulation makes the problem very sensitive to noise. A more convenient way to pose the problem consists in forcing $g(p_i)$ to be close to p_i. In this new formulation, the solution is less sensitive to noise and it is possible to restrict the class of deformations and still obtain a solution. The new problem is therefore an optimization problem.

Particularly interesting cases of the formulation above consist in considering space deformations induced by rigid motions, linear or projective transformations. A survey discussing several approaches to solve the problem using rigid motions can be seen in (Goodrich *et al.* , 1999). A comprehensive treatment of warping and morphing of graphics objects can be found in (Gomes *et al.* , 1998).

2.5.2 Image Analysis

A classic problem in the area of image analysis is edge detection. Intuitively, edges are the boundaries of objects in the image. This is shown in Figure 2.11, where the white pixels in image (b) indicate the edges of image (a).

(a) (b)

Figure 2.11: Edges of an image.

The first researcher to call attention to the importance of edge detection in image analysis was D. Marr[4]. He posed an important problem in this area which became known as the *Marr's Conjecture*: *an image is completely characterized by its edges*. In order to pose this problem more precisely it is necessary to define the concept of edge, but we will not do this here. We will resort, instead, to a high-level discussion of the problem.

Edge detection methods can be divided into two categories:

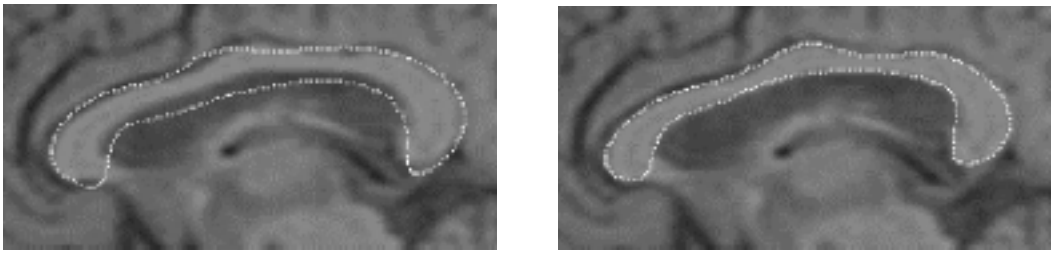
- Frequency based methods;
- Geometric methods.

[4]David Marr (1945-1980), English scientist and MIT researcher, was one of the pioneers in the area of computer vision.

Frequency-based methods characterize edges as regions of high frequency of the image (i.e., regions where the image function exhibit large variations). The theory of operators over images is used to determine those regions. The reconstruction of an image from its edges is therefore related with the invertibility of these operators. The wavelet transform plays here a significant role.

Geometric methods try to describe edges using planar curves over the image support. This approach employs largely optimization methods. A classic example is given by "snakes", that we will briefly describe next.

Snakes. These curves reduce the edge detection problem to a variational problem of curves in the domain of the image. We use the image function to define an external energy in the space of curves. This external energy is combined with an internal energy of stretching and bending, in order to create an energy functional. The functional is such that minimizer curves will be aligned with the edges of regions in the image. The computation of snakes is usually done using a relaxation process starting with a initial curve near the region. Figure 2.12 illustrates edge detection using snakes: (a) shows the initial curve; and (b) shows the final curve fully aligned with the edges of a region.



(a) (b)

Figure 2.12: Edge detection using snakes: (a) initial curve; (b) final edge curve (P. Brigger & Unser, 1998).

2.6 Animation and Video

The area of animation is concerned with the problem of time-varying graphics objects. There are important three problems in animation:

- Motion synthesis;
- Motion analysis;
- Motion processing.

In general, motion synthesis is a direct problem and motion analysis leads to various types of inverse problems.

Motion synthesis is closely related with the are of motion control and employs extensively optimization methods (e.g. optimal control).

Motion visualization, or animation, is done through a sequence of images, which constitute essentially a temporal sampling of the motion of the objects in a scene. Such an image sequence is also know as *digital video*. Note that this opens up a perspective for new problems: analysis and processing of video.

A particularly interesting problem consists in recovering the motion of 3D objects in the scene based on the analysis of their 2D motion in the video, which is known as *motion capture*. This is another example example of a highly ill-posed problem in the sense of Hadamard, where optimization methods are largely employed to find adequate solutions.

2.7 Classification of Graphics Objects

Classification problems allow the recognition of graphics objects based on the identification of the class to which they belong.

In the area of multimedia databases (image databases, for example) the classification methods play a fundamental role in the problem of indexation and query of objects in the database.

In this section we will give a generic description of the classification problem and will demonstrate how it leads to optimization methods in a natural manner.

Given a set of graphics objects $\mathcal{O}$, an *equivalence relation* is an association $\equiv$ between elements of $\mathcal{O}$ satisfying the following conditions:

1. $O_\alpha \equiv O_\alpha$, for all $O_\alpha \in \mathcal{O}$;
2. If $O_\alpha \equiv O_\beta$, then $O_\beta \equiv O_\alpha$;
3. If $O_\alpha \equiv O_\beta$ and $O_\beta \equiv O_\theta$, then $O_\alpha \equiv O_\theta$.

The equivalence relation $\equiv$ divides the set $\mathcal{O}$ of graphics objects into an union of disjoint sets:

$$\mathcal{O} = \bigcup_i \mathcal{O}_i, \quad \mathcal{O}_i \cap \mathcal{O}_j = \emptyset \quad \text{if} \quad i \neq j.$$

In fact, for each element $O_i \in \mathcal{O}$, we define $\mathcal{O}_i = \{O \in \mathcal{O}; O \equiv O_i\}$. This subdivision in disjoint parts is called a *partition*.

This partition provides us with a *classification* of the graphics objects of the set $\mathcal{O}$, where each set $\mathcal{O}_i$ of the partition defines a class. Any element can be taken as the representative of the class.

How to obtain classifications of families of graphics objects? A powerful method is to employ a *discriminating function*. That is, given a function $F\colon \mathcal{O} \to V$, where $V \subseteq \mathbb{R}$, we define an equivalence relation $\equiv$ in $\mathcal{O}$ taking

$$O_i \equiv O_j \quad \text{if, an only if,} \quad F(O_i) = F(O_j).$$

It is easy to verify that we have in fact an equivalence relation. In this case, each element $v \in V$ in the image of the function F, defines a equivalence class $F^{-1}(v) = \{O \in \mathcal{O}; F(O) = v\}$.

In general, when we have a classification of a set of objects, we want to choose an "optimal" representative, O_i in each class $\mathcal{O}_i$. For this we need to define a distance function [5] d between graphics objects in the space $\mathcal{O}$. Then, we choose the representative element O_i of each class $\mathcal{O}_i$ such that the function below is minimized

$$\sum_{O \in \mathcal{O}_i} d(O, O_i).$$

[5] A distance function differs from a metric because it does not necessarily satisfy the triangle inequality.

An interesting case is when the set $\mathcal{O}$ of graphics objects is finite, $\mathcal{O} = \{\mathcal{O}_1, \ldots, \mathcal{O}_m\}$. A classification amounts to divide $\mathcal{O}$ in a partition with n sets. $\mathcal{O}_1, \ldots, \mathcal{O}_n$, where $n < m$ (in general n much smaller than m). This problem can be naturally posed as an optimization problem: *Which is the optimal partition of the space $\mathcal{O}$?*

It is common in the classification problems to have a probability distribution p associated with the occurrence of each object in the set $\mathcal{O}$. In this case, a more suitable measure for an optimal partition would be

$$E = \sum_{i=1}^{m} \sum_{O \in \mathcal{O}_i} p(O) d(O, O_i).$$

The function E depends on the partition in classes and the choice of representative for each clas. The minimization of the function E over all possible partitions of the set $\mathcal{O}$ is a problem of combinatorial optimization. This problem is known in the literature as *cluster analysis*.

Example 9. ***(*Character recognition.)*** A classification problem of practical importance is the automatic character recognition: Given a binary image of a scanned text, identify all the printed glyphs of the text. This problem has applications in OCR ("optical character recognition") systems, which transform a scanned image in a text that can be edited in a word processor.

Note that each character represents a region of the plane, which can be interpreted as a planar graphical object. We need to identify the associated letter of the alphabet for each of these regions. If we define the depiction of a character as another planar graphics object, then the problem is to find a discriminating function to measure the similarity of this type of objects (Atallah, 1984). After that, we proceed as in the general classification problem described above.

Chapter 3

Optimization: an overview

In intuitive terms, optimization refers to the class of problems that consist in choosing the best among a set of alternatives.

Even in this simple, imprecise statement, one can identify the two fundamental elements of an optimization problem:

- *best choice*, that conveys establishing a criterium to choose the solution; this is usually expressed by means of an objective function to be minimized (or maximized);

- *alternatives*, that refers to the set of possible solutions, usually determined by a set of constraints – equalities and/or inequalities that must be satisfied by any candidate solution.

Therefore, an optimization problem may be posed as follows:

$$\min_{x \in S} f(x),$$

where S is the set of possible solutions and $f : S \to \mathbb{R}$ is the objective function. We may restrict attention to minimization problems, since any maximization problem can be converted into a minimization formulation, by just replacing f by $-f$.

3.1 Classification of Optimization Problems

Optimization problems can be classified according to several criteria, related to the properties of the objective function and of the set of solutions S. Thus, this classification will take into account:

- the nature of S;
- the nature of the constraints used to define S;
- the characteristics of the objective function f.

The main classification criterium is the nature of S, that leads to classifying problems as *continuous*, *discrete* or *combinatiorial*.

3.1.1 Continuous Problems

In this type of problem, S is "continuous". By that, we mean that S is a subset $\mathbb{R}^n$ determined by a finite (possibly empty) set of equalities and/or inequalities. Tipically, S is a manifold embedded in $\mathbb{R}^n$.

It is useful to consider here the infinite dimension case as well. In this case, the elements of S are functions and the optimization problem is called a *variational problem*.

3.1.2 Discrete Problems

In this type of problem, S is discrete (i.e., possesses no accumulation points). The most common cases are those in which $S \subseteq \mathbb{Z}^n$.

It is interesting to notice that some discrete problems may be solved more easily by first considering the same problem on the continuous domain.

Example 10. Consider the problem of maximizing $f(x) = 3x - 2x^2$ under each one of the restritions below:

i. $x \in \mathbb{R}$

ii. $x \in \mathbb{Z}$

The first case is trivial. Since f is a quadratic function and the coefficient of the leading term is negative, it has a maximum at the only m such that $f'(m) = 0$, which is equivalent to $3-4m = 0$. Thus, the maximum of f occurs at $m = \frac{3}{4}$.

The discrete case requires a more elaborate argument. Although the solution obtained for (i.) does not apply to (ii.), since $m \notin \mathbb{Z}$, we can use it to obtain the solution for (ii.). In this particular case, it suffices to choose the integer that is closest to m (1 in this case). This works because f is increasing on the interval $[-\infty, m]$, decreasing on $[m, \infty]$ and its graph is symmetric with respect to the line $x = m$. Thus, the farther x is from m, the smaller the value of $f(x)$. Therefore, $f(x)$ takes its maximum value over Z at the integer that is closest to m.

It is not always the case that the solution for the discrete version of an optimization problem derives so easily from the continuous version. For instance, the function $f(x) = x^4 - 14x$ has a point of minimum at $x \approx 1.52$. The integer that is closest to x is 2, where the value of f is -12. However, the minimum value of f over the integers occurs at 1, where the value of f is -13.

Thus, although solving the continuous version of a discrete problem is often used as part of the solution strategy, finding the solution on the discrete domain requires, in general, more than simply rounding the solutions found for the continuous case.

3.1.3 Combinatorial Problems

In this type of problem, the set S of all possible solutions is finite. In addition, usually the elements of S are not explicitly determined. Instead, they are indirectly specified through combinatorial relations. This allows S to be specified much more compactly than by simply enumerating its elements.

In contrast with continuous optimization, whose study has its roots in classical calculus, the interest in solution methods for combinatorial optmization problems is relatively recent and significantly associated with computer technology. Before computers, combinatorial optmization problems were less interesting, since they admit an obvious method of solution, that consists in examining all possible solutions in order to find the best one. The relative efficiency of the

possible methods of solution was not a very relevant issue: for real problems, involving sets with a large number of elements, any solution method, efficient or not, was inherently unfeasible, for requiring number of operations too large to be done by hand.

With computers, the search for efficient solution methods became imperative: the practical feasibility of solving a large scale problem by computational methods depends on the availability of a efficient solution method.

Example 11. This need is well illustrated by the *Traveling Salesman Problem*. Given n towns, one wishes to find the minimum length route that starts in a given town, goes through each one of the others and ends at the starting town.

The combinatorial structure of the problem is quite simple. Each possible route corresponds to one of the $(n-1)!$ circular permutations of the n towns. Thus, it suffices to enumerate these permutations, evaluate their lengths, and choose the optimal route.

This, however, becomes unpractical even for moderate values of n. For instance, for $n = 50$ there are approximatelly 10^{60} permutations to examine. Even if 1 billion of them were evaluated per second, examining all would require about 10^{51} seconds, or 10^{43} years! However, there are techniques that allow solving this problem, in practice, even for larger values of n.

3.2 Other classification criteria

There are other ways of classifying optimization problems, based on characteristics that can be exploited in the solution methods. This is especially useful for continuous problems.

3.2.1 Restrictions

As mentioned before, the set S of all possible solutions can be specified by means of the restrictions to be satisfied by its elements. In general, restrictions are expressed by conditions to be satisfied by certain functions $g_i(x)$, for $i = 1, \ldots m$. It is useful to classify problems with respect both to the *nature* of the

conditions imposed on those functions and to the *properties* of the functions themselves.

With respect to their nature, we have:

- equality restrictions: of the type $h_i(x) = 0$.
- inequality restricitions: of the type $g_j(x) \leq 0$.

S consists of the points that simultaneously satisfy all those restricitions, that is, that are in the intersection of the surfaces $h_i(x) = 0$ and the regions $g_j(x) \geq 0$. Equality and inequality restricitions are treated very differently by the algorithms used to solve continuous optimization problems.

Algorithms are also influenced by the properties of the functions used to express the restrictions. These properties can be exploited in devising specialized algorithms for certain classes of problems. The relevant cases include:

- linear functions;
- quadratic functions;
- convex (or concave) functions;
- sparse functions.

For instance, if all restrictions are of the form $g(x) \leq 0$, where g is convex, the set S of all possible solutions is convex. This can be exploited for obtaining special optimality conditions or algorithms.

3.2.2 Objective Function

The properties of the objective function f are also relevant for devising strategies to solve optimization problems. Among the special cases of f that can be exploited by the solution methods are:

- linear functions;
- quadratic functions;

- convex (or concave) functions;
- sparse functions;
- separable functions.

For instance, a widely studied situation is that in which we have a linear objective function together with restricitions expressed by linear equalities and/or inequalities. Optimization problems of this type are called *linear programs* and an entire body of techniques was developed exploiting these properties of the objective function and the restrictions.

3.3 Comments and Bibliographical References

The literature on the several aspects of optimization is very wide and a complete review is beyond the scope of these notes. However, we suggest below a few textbooks for a first contact with the area.

(Luenberger, 1984) gives a comprehensive view of continuous optimization, covering both linear and nonlinear problems. For discrete and combinatorial optimization, (Papadimitriou & Steiglitz, 1982) provides a good overview.

Many books specialize in linear programming and its applications to combinatorial problems (in particular, to network flow problems); (Chvatal, 1983) is a good example of such a book. (Fang & Puthenpura, 1993) is a more recent text, emphasyzing the relatively new interior point algorithms for linear programming. Another important reference is (Gill *et al.* , 1981), that tackles, mainly, nolinear optimization, with emphasys on the practical aspects concerning the choice of the algorithm appropriate for a given optmization problem.

Finally, (Weinstock, 1974) is an adequate reference for a first reading on variational optimization.

Chapter 4

Optimization methods

4.1 Continuous Optimization

Continuous optimization problems are of the form

$$\begin{array}{rll} \min & f(x) & \\ \text{subject to} & h_i(x) = 0, & i = 1, \ldots, m \\ & g_j(x) \leq 0, & j = 1, \ldots, p \end{array}$$

Most theorems and methods for continuous optimization deal with local optima, establishing necessary and sufficient conditions for local optimality and devising algorithms for reaching such a point.

4.1.1 Optimality conditions

The main result on optimality conditions is the following theorem.

Theorem 4.1. Karush-Kuhn-Tucker conditions (necessary first-order conditions) *Let x be a point of local miminum for the problem*

$$\begin{array}{rll} \min & f(x) & \\ \text{subject to} & h_i(x) = 0, & i = 1, \ldots, m \\ & g_j(x) \leq 0, & j = 1, \ldots, p \end{array}$$

and suppose that x_0 is a regular point for these restrictions (meaning that $\nabla f_i(x_0) \quad (i = 1, \ldots, n)$ and $\nabla g_j(x_0) \quad$ (for j such that $g_j(x_0) = 0$) are linearly independent). Then, there exists a vector $\lambda \in \mathbb{R}^m$ and a vector $\mu \in \mathbb{R}^p$, with $\mu \geq 0$, such that

$$\nabla f(x_0) + \sum_{i=1}^{n} \lambda_i \nabla h_i(x_0) + \sum_{j=1}^{p} \mu_j \nabla g_j(x_0) = 0$$

$$\sum_{j=1}^{p} \mu_j g_j(x_0) = 0$$

This generalizes the classical calculus theorems that treat the unrestricted case and the case where all restrictions are of the equality type (giving rise to the well-know theory of Lagrange multipliers).

Observe that the theorem above only provides necessary conditions for local optima (although it was stated for a minimization problem, the conditions are the same for maximization). There are other theorems that provide sufficient conditions for local maxima and minima, based on second-order derivatives (see (Luenberger, 1984) for details).

4.1.2 Quadratic problems

The optimality conditions in Theorem 4.1 establish a set of equalities and inequalities that must be satisfied for a candidate point in order to qualify as a possible extremum. In general, these equations and inequations are nonlinear and cannot be solved analytically. Thus, one has to resort to a iterative solution method, that produces a sequence of solutions that (hopefully) converge to a local optimum.

A notable exception is the family of problems of minimizing a quadratic objective function under linear equality constraints. That is, quadratic problems of the form:

$$\begin{aligned} \min \quad & c + b^T x + \tfrac{1}{2} x^T H x \\ \text{subject to} \quad & Cx = d, \end{aligned}$$

where b and c are $n \times 1$ vectors, H is a $n \times n$ nonnegative definite matrix, C is a $m \times n$ matrix (with $m < n$) , and d is a $m \times 1$ vector.

Problems of this form (called quadratic problems) have the following important properties:

- Since the objective function is convex and the set of feasible solutions is a convex set, any local minimum point is necessarily a global minimizer.

- The optimality conditions lead to a system of linear equations. Thus, they can be solved without resorting to iterative approximation methods. Moreover, all points that satisfy these conditions are indeed global minimizers.

An important special case of quadratic problem has objective function of the form $\min ||Ax - b||^2$ or, equivalently, $\min ||Ax - b||$. These are called *linear least-square* problems and occur frequently in applications when fitting a certain model for a set of observed data. Actually, this formulation applies to many inverse problems in computer graphics.

The study of quadratic problems is also important because many iterative methods are somehow inspired in the quadratic case. This is the case, for instance, of Newton's method and the conjugate gradient method (see (Luenberger, 1984) for details).

4.2 Combinatorial Optimization Methods

Combinatorial optimization methods achieve efficiency by exploiting special structure present in a given problem. This allows one to do substantially less work than performing exhaustive search on all possibilities. Although different strategies apply to different problems, there are some general paradigms that lead to the construction of efficient combinatorial algorithms for several problems at the same time.

One such approach is *dynamic programming*, also called *sequential* or *recursive* optimization. This method applies to combinatorial problems that can be decomposed in a sequence of *stages*. At each stage there is a finite set of

possible *states*. The basic operation at each stage consists in making the partial decision that leads to the optimal solution. This is based on the *Optimality Principle*, stated below.

Proposition 4.1 (Optimality Principle). *The best sequence of decisions has the property that, whatever the last decision might be, the sequence of decisions that leads to the decision before the last must also be optimal.*

Let $f_n(x)$ be the optimal cost for reaching state s at stage n. The optimality principle states that

$$f_{n+1}(x) = \min_{u \in \text{stage } n} \{f_n(x) + c(u, x)\} \tag{4.1}$$

where $c(u, x)$ is the transition cost from state u at stage n to state x at stage $n + 1$.

Equation 4.1 is known as the *Bellman equation* and provides a recursive way to find the optimal solution to a N-stage problem, by successively considering the optimal solutions for stages $1, 2, ..., N$ and for each possible state at those stages. The amount of work involved in doing the recursion typically is a small fraction of the effort that would be spent considering all possible solutions.

Dynamic programming formulations are closely related to finding the minimum cost path in a graph from a given origin s to a destination t. Actually, the formulation given in equation 4.1 corresponds to finding the minimum cost path in a graph where the vertices correspond to all possible pairs stage-state, edges connect vertices corresponding to consecutive stages, and the cost for each edge is the transition cost $c(u, x)$.

In the case where all edge-costs are nonnegative, the algorithm of choice for finding a minimum-cost path is Dijkstra's algorithm (see Algorithm 1), which runs in time $O(n^2)$, where n is the number of vertices of the graph. Dijkstra's algorithm can be seen as a dynamical programming formulation where the ordering of the vertices is not pre-determined. Instead, vertices are ordered during the algorithm, according to their distance to the origin (see citePapadimitriou for details).

Algorithm 1 Dijkstra's algorithm

1. **Initialization:**
   $c_1(1) = 0$ e $c_1(j) = a_{1j}$ for $j \neq 1$
   $p = \{1\}, \overline{p} = \{2, \ldots, |V|\}$
2. **Recursion:**

while $\overline{p} \neq \emptyset$ **do**
  Choose k shuch $c(k) = \min_{j \in \overline{p}} c(j)$
  $p = p \cup \{k\}$
  $\overline{p} = \overline{p} - \{k\}$
  for all $j \in \overline{p}$ **do**
    $c(j) = \min\{c_j(i), c(k) + a_{kj}\}$
  end for
end while

4.3 Solving optimization problems

In chapter 2, we saw several examples of problems in Computer Graphics naturally posed as optimization problems.

However, the simple writing as an optimization problem does not produce automatically a solution for the original problem.

As a matter of fact, optimization problems are usually hard to solve. Some of the difficulties are:

- In continuous optimization,the optimality conditions refer to local optima. Also, general solving procedures produce solutions that are only guaranteed to be local optima. Algorithms that search for global optima are usually very expensive. This leads us, in most cases, to accept the best local optimum produced by the solving algorithm.

- Most continuous optimization methods are iterative algorithms, requiring a good initial solution. In some cases, this initial solution may be hard to find.

- There are combinatorial problems – for example, the so-called *NP-complete* problems – for which there are no efficient methods (meaning running in polynomial time on the size of the instance to be solved) that provide exact solutions in all cases. The Traveling Salesman Problem, mentioned before,

is in that class. These problems are related in such a way that if a algorithm with polynomially-bounded running time is found for one of them, then polynomial time algorithms will be available for all of them. This makes it unlikely that such an algorithm will ever be found. When solving such problems, many times we have to use heuristic solution processes, that provide "good" (but not optimal) solutions.

- Many optimization problems in Computer Graphics are variational in nature (that is, the set of solutions has infinite dimension). Optimality conditions for such problems are expressed by partial differential equations, which cannot be solved analytically, in general. Therefore, solving such problems requires using numerical methods, which in turn require some discretization scheme. There are many choices for doing this discretization, as the following example illustrates.

  Example 12. Let us consider the problem of finding a minimum length curve joining points $P_1 = (x_1, y_1, f(x_1, y_1))$ and $P_2 = (x_2, y_2, f(x_2, y_2))$ on a surface in $\mathbb{R}^3$ having equation $z = f(x, y)$ (Fig 4.1).

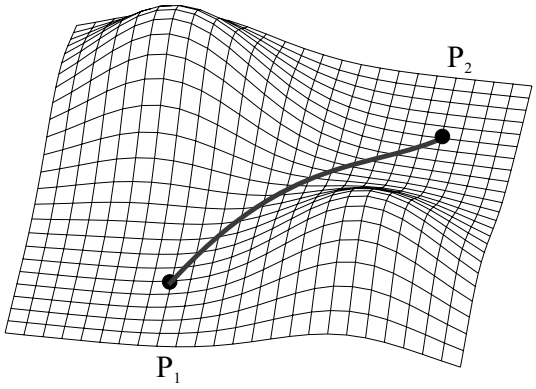


Figure 4.1: Minimum-length path on a surface

Similar mathematical formulations apply to problems that are apparently quite different in nature. For instance, consider the problem of detecting a segment of the boundary (between two given points) of the region corresponding to the brain, in the image in Figure 4.2. An image can be seen as a surface $z = f(x, y)$, where z is the gray-level at position (x, y).

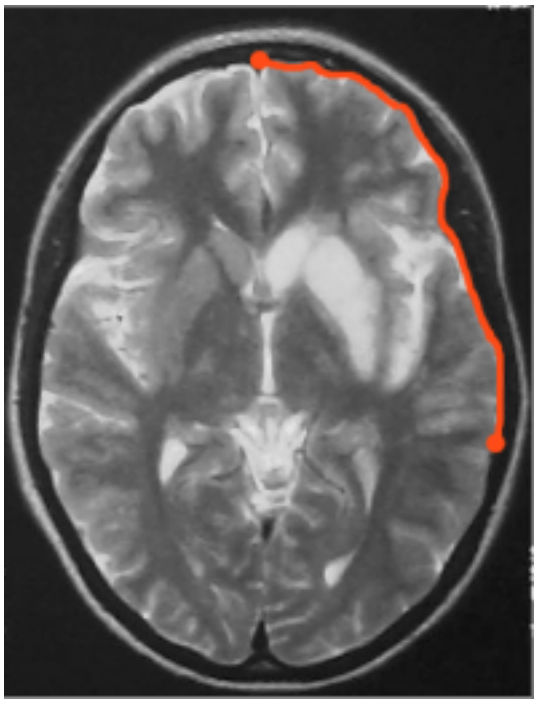


Figure 4.2: Detecting a boundary segment

The boundary segment can be found by minimizing an appropriate functional, of the form

$$\int_0^1 ||x'(t)||w(x(t))dt \tag{4.2}$$

over all curves $x(t)$ joining the two given points. The magnitude of the gradient tends to be large at points on the boundary. Thus, for the case at hand, choosing $w(x) = \frac{1}{1+|\nabla f(x)|}$ gives good results. Observe that the case of the minimum length curve is a particular instance of the formulation given in equation 4.2, for $w(x) = 1$.

In general, problems such as these cannot be solved analytically and require discretization. Choosing different stages in the solution process to discretize leads to different solution strategies. There are essentially three choices:

1. Attacking directly the variational problem

   This implies solving the Euler-Lagrange differential partial equations that express the optimality conditions for the problem. These cannot, in general, be solved analytically. Therefore, they must be solved numerically, which requires discretizing at this stage.

2. Approximating the problem by a continuous optimization problem (in finite dimension)

   This strategy consists in projecting the original infinite dimension space on a finite dimension space. For instance, we may consider the problem of mimimizing the length of curves parametrized by cubic polynomial functions of the form

   $$\alpha(t) = (c_{10} + c_{11}t + c_{12}t^2 + c_{13}t^3, c_{20} + c_{21}t + c_{22}t^2 + c_{23}t^3)$$

   This restricts our search to the space of dimension 8 of the parameters c_{ij}. Now, we can use continuous optimization methods for finding a solution. However, even in this simplified version of the problem, we must resort to discretization to compute approximations for the integral that yields the length of the curve.

3. Approximating the problem by a discrete optimization problem

   Here, all discretization occurs at the beginning of the solution process. The domain of f is discretized (normally using a rectangular grid). We then consider a weighted graph G in which the vertices are the nodes of the grid, the arcs correspond to the edges joining pairs of vertices, and the length of a given arc is the length of the curve on the surface generated by the corresponding edge (see Figure 3).

   The minimal length curve is found by solving a minimum-cost path problem in G. The quality of the approximation produced by this method depends, of course, on the resolution of the grid. Finer resolutions lead to better approximations. However, they also require a larger computational effort to solve the problem.

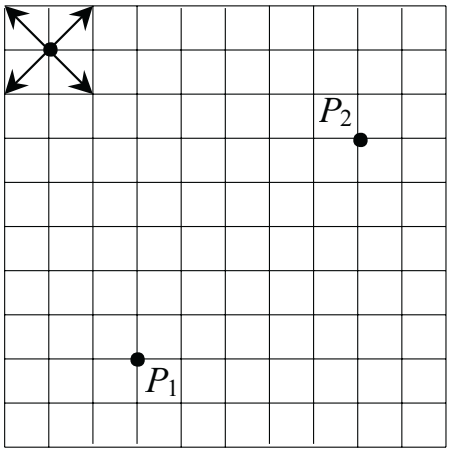


Figure 4.3: Regular discretization

Bibliography

Atallah, M. 1984. Checking similarity of planar figures. *International J. Comput. Inform. Sci.*, **13**, 279–290.

Chvatal, Vasek. 1983. *Linear Programming*. W. H. Freeman.

Fang, Shu-Cherng, & Puthenpura, Sarat. 1993. *Linear Optimization and Extensions: Theory and Algorithms*. Prentice Hall.

Gill, Philip E., Murray, Walter, & Wright, Margareth H. 1981. *Practical Optimization*. Academic Press.

Gomes, J., Darsa, L., Costa, B., & Velho, L. 1996. Graphical Objects. *The Visual Computer*, **12**, 269–282.

Gomes, Jonas, & Velho, Luiz. 1998. *Computação Gráfica, Volume 1*. Série de Computação e Matemática.

Gomes, Jonas, Darsa, Lucia, Costa, Bruno, & Velho, Luiz. 1998. *Warping and Morphing of Graphical Objects*. San Francisco: Morgan Kaufmann.

Goodrich, M. T., Mitchell, J.S. B., & Orletsky, M. W. 1999. Approximate Geometric Pattern Matching Under Rigid Motions. *PAMI*, **21**(4), 371–379.

Luenberger, David G. 1984. *Linear and Nonlinear Programming*. Addison-Wesley.

P. Brigger, R. Engel, & Unser, M. 1998. B-spline snakes and a JAVA interface: An intuitive tool for general contour delineation. *Proc. IEEE Int. Conf. on Image Processing, Chicago, IL, USA, October*, 4–7.

Papadimitriou, Christos H., & Steiglitz, Kenneth. 1982. *Combinatorial optimization: algorithms and complexity*. Prentice-Hall.

Weinstock, Robert. 1974. *Calculus of variations with apllications to physics and engineering*. Dover.