Reconstruction of a 3-D surface from its normal vectors

Reconstruction of a 3-D surface from its normal vectors

by

E. Zs. Lepoeter-Molnar

89/1

January, 1989

COMPUTING SCIENCE NOTES

This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science Eindhoven University of Technology.
Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review.
Copies of these notes are available from the author or the editor.

Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands

Editors: prof.dr.M.Rem
prof.dr.K.M. van Hee

RECONSTRUCTION OF A 3-D SURFACE FROM ITS NORMAL VECTORS

É. Zs. Lepoeter-Molnár

Endhoven University of Technology
Department of Mathematics and Computing Sience
P.O. Box 513
5600 MB Eindhoven, The Netherlands

December 1988

ABSTRACT

A well-known problem in pattern recognition is to determine the shape of a 3-D object from its 2-D image.

This paper describes a new, general algorithm to reconstruct a surface from its normal vectors relative to an arbitrary reference point. The depth values of the surface are calculated recursively, having smoothed the defect due to path dependency homogeneously. The theoretical background is worked out to prove the correctness of the program. The time complexity of the algorithm is given. Two applications of the general method are discussed with a detailed error analysis. The comparison of experimental results of this new approach and a previous one is presented. Theoretical analysis and experiments show that our method is both powerful and simple to implement.

Acknowledgement

I would like to thank Kees van Overveld for the valuable suggestions and Ágoston Eiben for his useful comments on a draft version of this paper. Photos made by Huub van de Wetering and implementation of the algorithm by Servaas van der Ploeg were much appreciated.

CONTENTS

1. Introduction
   1.1 The general shape from shading problem 3
   1.2 The reconstruction of the 3-D shape 3
2. Formulation of the problem
   2.1 Notational conventions 6
   2.2 A first statement of the problem 6
   2.3 Approximation of the Z values 6
   2.4 Conservativity 8
   2.5 Reformulation of the problem 11
3. Reconstruction algorithms
   3.1 The general algorithm 13
   3.2 Special cases of the approximation
      3.2.1 The quadratic method 16
      3.2.2 The cubic method 17
4. Error analysis
   4.1 Error of the digital integration 21
   4.2 The effect of random noise 23
   4.3 Some experiment of surface reconstruction 23
5. Conclusion 27
6. References 28

1. INTRODUCTION

1.1 The general shape-from-shading problem

Monochrome images of surfaces with homogeneous reflecting properties commonly exhibit a varying irradiance called *shading*. The shading is dependent on four principle factors : the illumination, the reflectance characteristics of the material, the shape of the surface, and the location of the observer (the viewpoint). The *shape-from-shading* problem is to extract the shape information from the irradiance data.

1.2 The reconstruction of the 3-D shape

A basic assumption is that the entire surface is visible. The problem then is to reconstruct the 3-D shape of the surface from its 2-D image. This reconstruction consists of the following steps:

1) separate the image of the surface from the background,
2) determine the local surface orientation as a *normal vector field*,
3) segment the image of the surface into regions called *figures* such that the normal vector field is continuous in every single region,
4) from the normal vector field derive the depth relative to some *reference point* on the surface.

Ad 1) The known technique of thresholding can be used [2].

Ad 2) There are additional assumptions :

i) the surface has homogeneous reflectance characteristics,
ii) its material is *isotropic* i. e. reflectance is not changed by rotating a surface element about an axis normal to the surface,
iii) the light source and the observer are distant from the surface. For computational convenience these distances are regarded as infinite.

Under these circumstances the radiance emanating from a small surface patch does not depend on its position in space. Furthermore, the perspective projection may be approximated by an orthogonal projection from the viewpoint.

With these assumptions the relationship between surface orientation and shading can be described by the *reflectance map* [1], which specifies the radiance as a function of surface orientation for fixed illumination, viewpoint, and reflectance characteristic. Since more normal vectors may map to the same irradiance value, the inverse mapping is not unique. A strategy to overcome this difficulty is discussed in Silver's work [2].

For practical applicability we also assume, that

iv) the variation in image irradiance is discrete and finite.

The reflectance map can be determined experimentally using a calibration object of known shape, e.g. a sphere of the same material and illuminated by the same light sources as the unknown surface.

Ad 3) Depth discontinuities can be recognized by contour detection technique [5], provided that they show themselves by orientation discontinuities (or cast shadows). Suggestions to find orientation discontinuities are described in section 5.

Ad 4) Let the surface be described by a function $Z(x,y)$. If this surface is once-differentiable, then an unnormalized normal vector at the 3-D point $(u,v,Z(u,v))$ can be written as $(Z_x(u,v),Z_y(u,v),-1)$. Z_x and Z_y stand for the first partial derivatives of Z with respect to x and y, respectively.

The Z value over some figure corresponding to a once-differentiable surface can be obtained by the integration :

$$Z(p) = Z(r') + \int_C \left(Z_x \, dx + Z_y \, dy \right)$$

along a curve C, from the projection r' of the given 3-D reference point to p.

There are several factors that may cause computational defects :

This integration behaves badly when the normal vector data is noisy. Then the value obtained at some point will depend on the integration path from r' to p.

In order, for the integral to be numerical computable the following assumption is necessary :

v) the image is digitized.

The numerical approximation of the integral from discrete values of the integrand also increases the path dependence of the result.

In order to make the outcome unique it is necessary to smooth away defects due to noise and discretisation.

The problem of this ambiguity of depth values has already been studied by Silver [2] and Wu and Li [4]. Their solutions consist of computing the Z values along two different paths and averaging. In this paper we propose an alternative method, where the error in depth estimation are distributed more homogeneously. This method turns out to give more stable results in some cases. Our algorithm has a guaranteed efficiency performance.

The purpose of this paper is to derive a general algorithm to perform step 4) of the reconstruction, e. i. to calculate a surface from its normal vector field.

In *section 2* the theoretical background of this algorithm is work out. In *section 3* the general algorithm is described with two special cases of the integral approximation. In *section 4* the error analysis and some experimental results are presented followed by concluding remarks in *section 5*.

2. FORMULATION OF THE PROBLEM

2.1 Notational conventions

Throughout of this paper, elements of $\mathbb{Z}^2$ are referred as *pixels*. They are mostly denoted by p,q or r.

The viewpoint is in the direction of the positive z-axes of a cartesian coordinate system. The surface is described by a function Z over some domain *G*, $G \subseteq \mathbb{R}^2$, of the image, $Z : G \to \mathbb{R}$.

The components of a 3-D point or a function f, $f : \Omega \to \mathbb{R}^3$, are denoted *f.x*, *f.y* and *f.z* respectively. For a constant *c*, $f \equiv c$ means that $f = c$ for all elements of its domain Ω.

C^n denotes the set of n times continuously differentiable functions.

2.2 A first statement of the problem

Formally the problem can be stated as follows. Given are

i) the digitized 2-D image of the surface :
a figure $\mathbf{F}$ with $\mathbf{F} \subseteq G \cap \mathbb{Z}^2$,

ii) a normal vector field $\mathbf{N}$, $\mathbf{N} : F \to \mathbb{R}^3$,

iii) a 3-D reference point $\mathbf{r}$ on the surface with $(\mathbf{r}.x, \mathbf{r}.y) \in \mathbf{F}$.

Assume that the normal vectors are given with $\mathbf{N}.z = -1$.

The problem may be regarded as that of calculating the values of a function Z over $\mathbf{F}$, $Z|_F$ such that

(2.2.1) $Z \in C^1$

(2.2.2) $Z_x|_F = \mathbf{N}.x$ and $Z_y|_F = \mathbf{N}.y$

(2.2.3) $Z(\mathbf{r}.x, \mathbf{r}.y) = \mathbf{r}.z$.

Notice that we restrict attention in (2.2.1) only to surfaces in C^1. This accords with our intuition on smoothness.

2.3 Approximation of the Z values

Obviously any primitive function of a function $\mathbf{n}$ ($\mathbf{n}$ is a continuous extension of $\mathbf{N}$) with

a) $\mathbf{n} : G \to \mathbb{R}^2$,

b) $\mathbf{n} \in C$,

c) the integral of $\mathbf{n}$, computed along a path within G, does not depend on this path[1],

d) $\mathbf{n}|_F = (\mathbf{N}.x, \mathbf{N}.y)$

satisfies the requirements (2.2.1) and (2.2.2). When adding an appropriate constant to a primitive function, (2.2.3) will hold as well.

If the normal vector field $\mathbf{N}$ belongs to a real 3-D C^1 surface, a function $\mathbf{n}$ exists satisfying a), b), c) and d), thus also the required primitive function exists.

The integral of the Lagrange interpolating polynomial l with ($\forall\ p : p \in F : l(p) = (\mathbf{N}.x(p), \mathbf{N}.y(p))$) could satisfy all the requirements (2.2.1)-(2.2.3). Since the computation of this polynomial is complicated, it oscillates strongly and the solution is numerically not stable[2], we choose an alternative solution to approximate a required primitive function. Our approach will be algorithmically efficient and numerically stable. In order to describe this approximation some definitions are needed.

DEFINITION 2.3.1 For $\Delta \in \mathbb{Z}^+$ p and q are *Δ-neighbours* iff

$$|p.x-q.x| = \Delta \wedge p.y = q.y \vee |p.y-q.y| = \Delta \wedge p.x = q.x.$$

DEFINITION 2.3.2 For Δ-neighbour pixels p,q of figure $\mathbf{F}$ a *weight function* w.r.t. the normal vector field $\mathbf{N}$ is $w_N : \mathbf{F} \times \mathbf{F} \to \mathbb{R}$ with the property : $w_N(p,q) = -w_N(q,p)$.

For a given value of Z(p) we approximate the sequential Z value in q by a general integration formula:

$$Z(q) = Z(p) + w_N(p,q) \qquad (2.1)$$

[1] If in (2.2.1) C^2 is required instead of C^1, then c) holds iff $n.x_y = n.y_x$.

[2] For example, continuous functions exist such that the integral of of the Lagrange interpolating polynomial of degree n does not tend to the original integral, when n approaching to infinity [7].

Note that a choice of the weight function corresponds to a choice of a numerical integration formula :

EXAMPLE 2.3.1 If we choose the weight function w.r.t. **N** for Δ-neighbour pixels p,q of **F** as

$$w_N(p,q) := (p.x-q.x)\,\frac{\mathbf{N}.x(p)+\mathbf{N}.x(q)}{2} + (p.y-q.y)\,\frac{\mathbf{N}.y(p)+\mathbf{N}.y(q)}{2}$$

the well known trapezoidal rule is obtained in the approximation (2.1) :

$$Z(q) = Z(p) + \begin{cases} \dfrac{\mathbf{N}.x(p) + \mathbf{N}.x(q)}{2} \cdot \Delta & \text{if } p.y=q.y \\ \dfrac{\mathbf{N}.y(p) + \mathbf{N}.y(q)}{2} \cdot \Delta & \text{if } p.x=q.x \end{cases}$$

EXAMPLE 2.3.2 An other other choice of the weight function w.r.t. **N** in the case of $\Delta = 2$ and $p.x \le q.x \wedge p.y \le q.y$ could be:

$$w_N(p,q) = \begin{cases} (q.x-p.x)\dfrac{\mathbf{N}.x(p) + 4\cdot\mathbf{N}.x(p.x+1,p.y) + \mathbf{N}.x(q)}{6} & \text{if } p.y=q.y \\ (q.y-p.y)\dfrac{\mathbf{N}.y(p) + 4\cdot\mathbf{N}.y(p.x,p.y+1) + \mathbf{N}.y(q)}{6} & \text{if } p.x=q.x \end{cases}$$

The formula (2.1) yields the Simpson's rule in this case :

$$Z(q) = Z(p) + \begin{cases} \dfrac{\mathbf{N}.x(p) + 4\cdot\mathbf{N}.x(p.x+1,p.y) + \mathbf{N}.x(q)}{3} & \text{if } p.y=q.y \\ \dfrac{\mathbf{N}.y(p) + 4\cdot\mathbf{N}.y(p.x,p.y+1) + \mathbf{N}.y(q)}{3} & \text{if } p.x=q.x \end{cases}$$

In these approximations the integrand satisfies a), b) and d). Property c) can not be guaranteed generally by special numerical integration formulas because of the approximation error [6] and eventual noise in the normal vector data.

2.4 Conservativity

In order to approximate the Z value from a known value in p also for a not Δ-neighbour pixel q, the locally used integration rule (2.1) can be extended. The full integral is approximated by

the sum of the approximations to the sub integrals taken along a path from p to q :

DEFINITION 2.4.1 A *path* π from p to q (abbreviated as $\pi(p,q)$) is a sequence of Δ-neighbour pixels, $r_0,\ldots,r_n$: $r_0 = p \wedge r_n = q$. The *length of* π *denoted* by $|\pi|$ is n, *begin*(π) = p and *end*(π) = q. $\langle\pi\rangle$ denotes the set of pixels of this path.
If Π is a set of path, $\langle\Pi\rangle = \bigcup_{\pi \in \Pi} \langle\pi\rangle$.
For a set V, $V \subseteq \mathbb{Z}^2$, the following notation is used :

$$\Pi_V = \{ \pi : \pi \text{ is a path} \wedge \langle\pi\rangle \subseteq V \}.$$

DEFINITION 2.4.2 The *weight* w.r.t. **N** *of a path* π entirely within **F** is defined by

$$W_N(\pi) := \sum_{i=1}^{|\pi|} w_N(r_{i-1},r_i).$$

With these notations the extended integration rule (2.1) :

$$Z(q) = Z(p) + W_N\big(\pi(p,q)\big) \qquad (2.2)$$

Unfortunately the numerical integration (2.2) taken along different paths between two pixels may not agree on the Z value. Formally the idealistic case, when the Z value does not depend on the chosen path (thus the Z values can be determined uniquely), is :

DEFINITION 2.4.3 Let Π be a set of paths in **F**, w_N a weight function w.r.t. **N**. The normal vector field **N** of figure **F** is *conservative* w.r.t. w_N and Π, denoted by *cons*$(\mathbf{N},\mathbf{F};w_N,\Pi)$, iff for any $p,q \in \mathbf{F}$ and any paths $\pi_1,\pi_2 \in \Pi$ with begin(π_1)=begin(π_2)=p and end(π_1)=end(π_2)=q : $W_N(\pi_1) = W_N(\pi_2)$.

For the correctness of the general algorithm (in section 3.1) we can rather use a local property of $\mathbf{N}$, $\mathbf{F}$ and w_N, which is equivalent with conservativity :

DEFINITION 2.4.4 A path π is *closed* iff begin(π) = end(π). For $\Delta \in \mathbb{Z}$ a closed path is called Δ-square iff it is composed of four different pixels and its length is 4.
For a set of Δ-squares S the following notation is used :

$$\langle S \rangle = \bigcup_{s \in S} \langle s \rangle$$

$S_V = \{ s : s \text{ is } \Delta\text{-square} \wedge \langle s \rangle \subseteq V \}$, where $V \subseteq \mathbb{Z}^2$.

DEFINITION 2.4.5 The normal vector field $\mathbf{N}$ of figure $\mathbf{F}$ is *locally conservative* w.r.t. w_N and s, $s \subseteq S_F$, iff cons($\mathbf{N},\mathbf{F};w_N,s$).

LEMMA 2.4.1 For any weight function w.r.t. $\mathbf{N}$, w_N, and any subset s of S_F:

$$\text{cons}\ (\mathbf{N},\mathbf{F};w_N,\Pi_{\langle s \rangle}) \quad \text{iff} \quad \text{cons}\ (\mathbf{N},\mathbf{F};w_N,S_{\langle s \rangle}).$$

DEFINITION 2.4.6 A *rectangle* R(a,c) is a subset of $\mathbf{F}$:
$R(a,c) = \{ p : a.x \le p.x \le c.x \wedge a.y \le p.y \le c.y$
$\wedge\ (p.x-a.x) \bmod \Delta = (p.y-a.y) \bmod \Delta = 0 \}$,
where $a,c \in \mathbf{F}$ and $(c.x-a.x) \bmod \Delta = (c.y-a.y) \bmod \Delta = 0$ too.
Its *boundary* B(a,c) is the shortest closed path from a that contains (c.x,a.y) and c and (a.x,c.y) in this order.

Formally, the difference in Z value, that is obtained applying rule (2.2) along a closed path (for algorithmic simplicity along a boundary of a rectangle, as we can see in section 3.1), is defined on the following way :

DEFINITION 2.4.7 The *defect* of boundary B(a,c) of a rectangle R(a,c) w.r.t. w_N is $D_{W_N}(a,c) := W_N(B(a,c))$.

The basic idea of the proof of lemma 2.4.1 is very simple using this notion :

PROPERTY 2.4.1 Let $a,c \in \mathbb{F}$ and R(a,c) a rectangle. If $c' \in R(a,c)$ with $c'.y = c.y \wedge (c'.x - c.x) \bmod \Delta = 0$ (as in fig. 2.4.1) then

$$D_{W_N}(a,c') + D_{W_N}((c'.x,a.y),c) = D_{W_N}(a,c) \ .$$

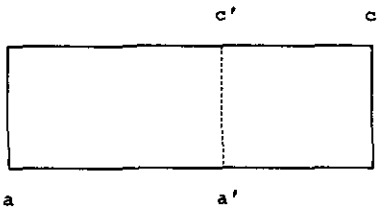


FIGURE 2.4.1

Obviously this result is the same if we split the rectangle horizontally.

2.5 Reformulation of the problem

To deal with the above mentioned inconsistency in the Z values, the approach of the problem could be modified. A correction term can be added to the weight function to suppress this inconsistency. This means that a discrete function is added to a function, which satisfies the requirements (2.2.1) and (2.2.2).

We need an additional definition to describe this method formally :

DEFINITION 2.5.1 A *distribution* corresponding to the boundary of a rectangle R(a,c), denoted by $f^{a,c}$, is an $|B(a,c)|$-tuple with

$$(\ \forall\ i:\ 0 \le i < |B(a,c)|\ :\ 0 \le f_i^{a,c}\ )\ \wedge \sum_{i=0}^{|B(a,c)|-1} f_i^{a,c} = 1.$$

($f^{a,c}(r_i,r_{i+1})$ is abbreviated as $f_i^{a,c}$.)

PROPERTY 2.5.1 Let w_N be a weight function w.r.t. **N**, R(a,c) a rectangle with the boundary $r_0,\ldots,r_n$ and $f^{a,c}$ be a distribution. Modify w_N on B(a,c) as

$$w_N^{f^{a,c}}(r_i,r_{i+1}) = w_N(r_i,r_{i+1}) - f_i^{a,c} \cdot D_{w_N}(a,c) \quad \text{for } i = 1,\ldots,n.$$

Then the following holds :

$$\sum_{i=0}^{n-1} w_N^{f^{a,c}}(r_i,r_{i+1}) = 0.$$

Note that $\mathrm{cons}(\mathbf{N}, \mathbf{F} ; w_N^{f^{a,c}}, \Pi_{<B(a,c)>})$ holds and the following is satisfied :

PROPERTY 2.5.2

$$w_N^{f^{a,c}} = w_N|_{<B(a,c)>} \quad \text{iff} \quad \mathrm{cons}\ (\mathbf{N},\mathbf{F};w_N,\Pi_{<B(a,c)>}).$$

The reformulated problem now is to calculate $Z|_F$ such that $Z|_F = Z_c|_F + Z_d$, where :

(2.5.1) $Z_c \in C^1 \wedge Z_d : \mathbf{F} \to \mathbb{R}$

(2.5.2) $Z_c|_F = \mathbf{N}.x \wedge Z_c|_F = \mathbf{N}.y$

(2.5.3) $Z(\mathbf{r}.x,\mathbf{r}.y) = \mathbf{r}.z$

(2.5.4) $Z|_F$ is uniquely determined independently on the path of the reconstruction from $\mathbf{r}'$ to any p, $p \in \mathbf{F}$, w.r.t. a set of distributions.

Ad (2.5.4) Assume that a set of distributions is also given corresponding to the figure **F** : $\mathcal{F}_F = \{ f^{a,c} : B(a,c) \subseteq \mathbf{F} \}$ such that for all Δ-neighbours p and q, that belong to the boundary of a rectangle within **F**

$(\# (a,c) : p,q \in <B(a,c)> \wedge <B(a,c)> \subseteq \mathbf{F} : f^{a,c}(p,q) \neq 0) = 1.$

With this property of $\mathcal{F}_F$ the Z values are modified only once in the recursive algorithm (see in section 3.1).

REMARK 2.5.1 Note that the C^1 part of function Z, Z_c can be obtained by the integration formula (2.2) with the weight function w_N. To compute Z we need to use the modified one, $w_N^{f^{a,c}}$.

3. RECONSTRUCTION ALGORITHMS

3.1 The general algorithm

The Z values are required :

i) over the figure **F**,

ii) from the normal vector field **N** ,

iii) beginning with the reference point **r**,

using a weight function w_N, which is consistent with the required type of Z_c in the numerical integration formula (2.2). A set of distribution $\mathfrak{F}$ is also given in order to calculate with a modified weight function for every a and c, where $\langle B(a,c)\rangle \subseteq \mathbf{F}$ (denoted by w_N^f).

For simplicity we shall restrict our consideration to a rectangle as figure **F** : **F** = R(A,C) for constants A,C, $A,C \in \mathbb{Z}^2$, with reference point **r**.x = A.x and **r**.y = A.y.

A recursive algorithm is presented for figure R(A,C) :

First the defect of the boundary B(A,C) is obtained. Then we calculate Z from the last known value, at the beginning this is **r**.z, using integration formula (2.2) with $w_N^{f^{A,C}}$. From property (2.5.1) the boundary is conservative w.r.t. $w_N^{f^{A,C}}$. After these steps we split the rectangle in two parts and continue the above described algorithm recursively. This approach is continued until reaching the Δ-squares in the rectangle.

As follows from lemma 2.4.1, ultimately the Z values correspond to **N** with cons(**N**,R(A,C);w_N^f,$\Pi_{R(A,C)}$).

```
PROCEDURE Reconstruction
(N : array [R(A,C)] of 2-D vector;
 z0 : real;
 st : integer;
 function w(p,q : pixel) : real;
 Φ : set of function f(a,c : pixel) : array [0..|B(a,c)|-1] of real;
 var Z : array [R(A,C)] of real);
```

```
{precondition: A,C ∈ Z² ∧ A.x < C.x ∧ A.y < C.y ∧ R(A,C) = F
               ∧ ( ∀ p : p ∈ R(A,C) : N(p) = (N.x(p),N.y(p))
               ∧ A = (r.x,r.y) ∧ z_0 = r.z
               ∧ st = Δ
               ∧ w = w_N for Δ neighbours p and q
               ∧ Φ = Ψ
 postcondition : ( ∀ p : p ∈ R(A,C) : Z value is uniquely
                         determined by (2.2) with w_N^f) }

procedure Correction( a,c: pixel; P: path; D: real );
          {precondition :
           A.x ≤ a.x < c.x ≤ C.x ∧ A.y ≤ a.y < c.y ≤ C.y
           ∧ <P> ⊆ <B(a,c)>
           ∧   Z|_{< B (a,c)>\(<P>\begin(P)\end(P))>}  is  known
           ∧ P'= Π( end(P),begin(P) ) ∧ <P'>∪<P> = <B(a,c)>
           ∧ f^{a,c}|_{p,} = 0
           ∧ D = D(a,c)
           postcondition :
           ( ∀ p : p ∈ R(a,c) : Z value is uniquely determined
                  by (2.2) with w_N^f ) }

          var i : integer;
              D_1,D_2,d : real;
              a',c' : pixel;
              P_1,P_2 : path;
          function Dsum( P: path ) : real;
                   {precondition : <P> ⊆ <B(a,c)>
                    postcondition : Dsum = W_N(P) }
  begin {Correction}
      if  |P| ≠ 0  →
        for i := 0 to |P|  →
            Z(r_{i+1})  := Z(r_i) + w(r_i,r_{i+1}) - D · f(a,c)[i];
                    { num. integration (2.2) with w_N^{f a,c} (p,q) }
        rof ;
      fi;
```

```
{ ( ∀ p : p ∈ <B(a,c)> : Z value is uniquely
                          determined by (2.2) with w_N^{f^{a,c}} ) }
  if  R(a,b) ≠ Δ square  →
      if  c.x-a.x > c.y-a.y  →
          a' :≡ (a + (c.x,a.y)) div 2 ;
          c' :≡ ((a.x,c.y) + c) div 2 ;
          { D(a,c') = Z(a') - Z(c') + Dsum( P(a',c') )
            ∧ D(a',c) = Z(c') - Z(a') - Dsum( P(a',c') ) }
      □   a' :≡ (a + (a.x,c.y)) div 2 ;
          c' :≡ ((c.x,a.y) + c) div 2 ;
          { D(a,c') = Z(c') - Z(a') - Dsum( P(a',c') )
            ∧ D(a',c) = Z(a') - Z(c') + Dsum( P(a',c') ) }
      fi ;
      d := Dsum( P(a',c') ) ;
      D1 := Z(a') - Z(c') + d;  P1 :≡ P(a',c');
      D2 := Z(c') - Z(a') - d ; P2 :≡ P(c',a');
      if c.x-a.x ≤ c.y-a.y  →  Swap(D1,D2);Swap(P1,P2)  fi;
      { The precondition of Correction holds }
      Correction( a, c', P(a',c'), D1 ) ;
      Correction( a', c, P(c',a'), D2 ) ;
   fi;
end {Correction};

begin {Reconstruction}
    Z(A) := z0 ;     { Z(A)=r.z }
    Correction( A, C, B(A,C), D(A,C) ) ;
end {Reconstruction};
```

Readers that are not familiar with the guarded command language, are referred to [8]. :≡ stands for element-wise or component-wise assignment.

Memory requirement of this algorithm looks space-consuming because of the general data type of Φ. But in the implementation (see in section 4) we use special distributions that are simpler to implement than a function of arrays. These variables require much less memory.

Since for each Δ-neighbour pixels p and q in R(A,C), w(p,q) may have to be modified :

PROPERTY 3.3.1

The above program has time complexity : $O(|R(A,C)|)$.

3.2. Special cases of the approximation

In this section our method will be applied to construct some special kind of surfaces assuming that

(3.2.1) $Z_c \in C^1 \wedge Z_c \in P_{i,j}(x,y)$ for each square with side Δ,

where $P_{i,j}(x,y)$ is the set of polynomials whose degree does not exceed i in variable x and j in variable y.

Note that this is a special case of (2.5.1) for Z_c.

3.2.1 The quadratic method

Assume that $i = j = 2$ in (3.2.1).

Z_c is chosen as a piece-wise biquadratic polynomial in C^1, hence it can be obtained as a primitive function of a piece-wise bilinear continuous function. The integral of a linear function is exactly computed by the trapezoidal rule. Therefor in the approximation (2.1) we apply the weight function as in example 2.3.1. Since two points determine a linear function and all the information about the normal vector field is worth-while to use, we define $\Delta := 1$.

The approximation error of this numerical integration is proportional to the second derivative of the integrand, thus the following holds:

PROPERTY 3.2.1 If a polynomial ρ exists, $\rho \in P_{2,2}$ satisfying $\rho_x|_F = \mathbf{N}.x \wedge \rho_y|_F = \mathbf{N}.y$ then

i) $cons(\mathbf{N},\mathbf{F};w_N,\Pi)$ for the above weight function w_N and any set of paths Π in $\mathbf{F}$

ii) ρ is defined by the integration formula (2.2) with w_N beginning at $Z(\mathbf{r}.x,\mathbf{r}.y)$.

The consequence of i) is : $Z_d \equiv 0$. ii) means that the reconstruction of a biquadratic surface is exact.

The general algorithm described in section 3.1 can be directly applied to obtain a piece-wise biquadratic C^1 polynomial with $\Delta = 1$ and weight function w_N as in example 2.3.1 .

3.2.2 The cubic method

Assume that i = j = 3 in (3.2.1).

Z_c is chosen to construct as a piece-wise bicubic polynomial in C^1, hence it is a primitive function of a piece-wise continuous biquadratic function. The integral of a quadratic function is exactly calculated by the Simpson's rule. Therefor we apply the weight function as in example 2.3.2 in the numerical integration (2.1). Since three points determine a parabola, we define $\Delta = 2$.

The approximation error is proportional to the 4th derivative of the integrand. Hence the following is satisfied:

PROPERTY 3.2.2 If a polynomial ρ exists, $\rho \in P_{3,3}(x,y)$ satisfying $\rho_x|_F = \mathbf{N}.x \wedge \rho_y|_F = \mathbf{N}.y$ then:

i) $\mathrm{cons}(\mathbf{N},\mathbf{F};w_N,\Pi)$ for the above mentioned weight function w_N and any set of paths Π in figure $\mathbf{F}$

ii) ρ is defined by the integration formula (2.2) with w_N beginning at the value of $Z(\mathbf{r}.x,\mathbf{r}.y)$.

Analogously to the quadratic method a bicubic surface can be exactly reconstructed by approximation (2.2) with the above mentioned w_N and $Z_d \equiv 0$ holds.

Note that conservativity, however a necessary condition for exact integration by approximation (2.2), of course is not sufficient:

PROPOSITION 3.2.1 For a weight function w_N, corresponding to an exact construction of a piece-wise polynomial surface of degree (n,m) in (2.2) and for any set of paths Π in $\mathbf{F}$:

$$\mathrm{cons}(\mathbf{N},\mathbf{F};w_N,\Pi)$$

if a function f exists, $f: G \to \mathbb{R}$ with $f = \rho + g + h$, satisfying $f_x|_F = \mathbf{N}.x \wedge f_y|_F = \mathbf{N}.y$, where $\rho \in P_{m,n}(x,y)$, g and h arbitrary C^1 functions and g depends solely on variable x, h on variable y.

In this case Z is obtained by the approximation (2.2) as a $P_{m,n}$ surface (this surface is not the above mentioned polynomial ρ, if $g \not\equiv 0$ or $h \not\equiv 0$). If $g,h \notin P_{m,n}$ and one of them is not identically 0 then the reconstruction of the surface f is not exact in spite of $Z_d \equiv 0$ holds.

Since $\Delta = 2$, we have to modify the program in section 3.1 to get the Z value also of the intermediate pixels belonging to the chosen type of the C^1 surface. The approximation of the Z values of the internal pixels is the value of the cubic polynomial, denoted by ρ_3, uniquely determined by the derivative in three consecutive points and its value one of the end pixels :

function *Intvalue*(val,d_0,d_1,d_2: real;forw : boolean) : real;
{precondition : val = $\rho_3(u)$ if forw, or val = $\rho(u+2)$ else
for an $u \in \mathbb{Z}$
$\wedge\ d_0 = \rho_3'(u)\ \wedge\ d_1 = \rho_3'(u+1)\ \wedge\ d_2 = \rho_3'(u+2)$
postcondition : Intvalue = $\rho_3(u+1)$ }

The internal Z value could be calculated by modified values for the derivative, according to the difference between the begin and end Z value of the interval.

With function Intvalue and extra variables m, f_1,f_2 and f_3 with $f_1 + 4 \cdot f_2 + f_3 = 1$ of type real (m is used for the sake of brevity), the algorithm Correction can be completed by the following statement :

```
m := D · f(a,c)[i] ;
if  r_i.x = r_{i+1}.x  →
    Z( r_i.x + 1,r_i.y ) := Intvalue( Z(r_i), N.x(r_i) - m · f_1,
                                      N.x(r_i.x+1,r_i.y) - m · f_2,
                                      N .x(r_{i+1})-m·f_3,  r_i.x <r_{i+1}.x );
fi ;
```

The statement for N.y can be modified in a similar manner.

We refer this procedure as *Correction-Inter.*

Denote R_1 and R_2 those points, where Z is calculated from A and A+$\underline{1}$ respectively having executed Correction-Inter. $\underline{1}$ stands for vector (1,1).

R_1:= [A.x .. C.x] × [A.y .. C.y] \ { p : p.x-A.x mod 2 = 1
$\wedge$ p.y-A.y mod 2 = 1 },

$R_2 := [A.x+1 \,..\, C.x-1] \times [A.y+1 \,..\, C.y-1] \setminus \{ p : p.x-A.x \bmod 2 = 0 \wedge p.y-A.y \bmod 2 = 0 \}$

Com denotes the points with "doubly" calculated Z value :

$Com := R_1 \cap R_2$.

In the following example R_1 and R_2 are drawn by single and double line respectively :

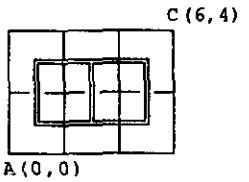


FIGURE 3.2.1

For R_2, the rest of the pixels, Z can be constructed by the algorithm Correction-Inter beginning with the reference value 0. Then the new Z "surface is translated" such that the average of the new values of pixels in Com (whose values have already been calculated from the reference point **r**), is equal to the average of their first Z values. At points in Com (where also a new value is obtained), we can choose this value or the old one or average of them as ultimate Z value.

For the "translation of the surface" the function *Average* is used :

function *Average*(z : array[Com] of real) : real;

{postcondition : $\text{Average} = \sum_{q \in Com} \frac{z(q)}{|Com|}$ }

The modified body of the program to calculate Z over [A.x .. C.x] × [A.y .. C.y] with a piece-wise bicubic C^1 part, with this completed procedure Correction-Inter, array Z_1[Com] of real and function Average is :

```
begin { Reconstruction of a piece-wise bicubic surface }
      Z(A) := z_0 ;
      Correction-Inter( A, C, B(A,C), D(A,C) );
      {calculation over R_1}
      Z_1 :≡ Z|_Com ;      { Z_1 = the first calculated value over Com }
      Z( A+1 ) := 0 ;
      Correction-Inter( A+1, C-1, B ( A+1, C-1 ), D ( A+1, C+1 ) ) ;
      { calculation also over R_2 }
      Z|_R2 :≡ Z|_R2 + Average( Z|_1 - Z_Com ) ;
     {"translation of the surface" above R_2 }
     Z|_Com :≡ (Z|_Com + Z_1) / 2 ;
     { average of the doubly calculated values }
end {Reconstruction of a piece-wise bicubic surface} .
```

4. ERROR ANALYSIS

The error in the depth reconstruction is caused by two factors in the above mentioned algorithm : the error of the digital integration and the error of the surface normal estimates.

To analyze the effect of these errors, a numerical simulation has been performed. This simulation was band on a perturbation of the normal vector field. The error corrupted normal vector field is simulated by adding a noise term according to :
$N.x = \varphi_x|_F + n$ and $N.y = \varphi_y|_F + n$, where φ is a known surface, n is a random noise with expected value 0 : $E[n] = 0$. The error is examined for $p \in F$:

$$e_n(p) = Z(p) - \varphi(p) \ .$$

In order to spread the defect homogeneously, uniform distribution is used on path P in procedure Correction (see in section 3.1) : $f^{a,c}|_P \equiv \frac{1}{|P|}$.

In this section some extra notations are used :
$d_\infty(v_1,v_2) = \max (|v_1.x-v_2.x| , |v_1.y-v_2.y|)$ for $v_1,v_2 \in \mathbb{R}^2$,
$\| f \|_G = \max_{x \in G} (f(x))$ for a function f, $f : G \longrightarrow \mathbb{R}$,
$R = [A.x,C.x] \times [A.y,C.y]$
and the resolution is $M \times M$. ($M = d_\infty(A,C)$ and for simplicity we usually choose M as 2^n for an $n \in \mathbb{N}$.)

4.1 Error of the digital integration

If there is no noise, the obtained error in the algorithm is due to the numerical integration only :

4.1.1 LEMMA Suppose that $n \equiv 0$, $R(a,c) \subseteq F$ and belongs to the kth level of the recursion with $k = \begin{cases} 2i & \text{if k is even,} \\ 2i+1 & \text{if k is odd,} \end{cases}$ then for any p, $p \in \langle B(a,c) \rangle$:

$$|e_0(p)| \leq \left(|\pi^*(r',p)| + \begin{cases} 2^i & \text{if } i = 0 \\ 2^i \cdot i & \text{if } i \neq 0 \end{cases} \right) \cdot c,$$

where π^* is the path on the boundary of the recursion rectangles in the order of the algorithm (r' is the projection of the reference point), and

$$c = \begin{cases} \dfrac{\| z^{(3)}(p) \|_R}{12} & \text{applying the quadratic method} \\ \dfrac{\| z^{(5)}(p) \|_R \cdot 4}{45} & \text{applying the cubic method .} \end{cases}$$

From this lemma we have that for all $p \in F$:

$$|e_0(p)| \le M \cdot (4 + \log M) \cdot c .$$

This lemma is not difficult to prove with the aid of the following recursive formulas :

$$|e_0(p_0)| \le (|\pi^*(r', p_0)| + 1) \cdot c \qquad (4.1)$$

$$|e_0(p)| \le |e_0(q)| \cdot \left(1 + \frac{2^i}{M} \right) + |e_0(s)| \cdot \frac{2^i}{M} + \left(\frac{2^{i+1}}{M} + 1 \right) \cdot c \qquad (4.2)$$

where p_0 stands for the splitting points of R(A,C) : for a_{2i}, c_0 and c_{2i+1} (belonging to the kth reconstruction level). $p = \frac{q+s}{2}$, q and s are splitting points of of the rectangle belonging to the previous recursion level :

$p = a_{2i+1}$ if $q = a_{2i}$ and $s = c_{2i}$ or

$p = c_{2i}$ if $q = a_{2i-1}$ and $s = c_{2i-1}$:

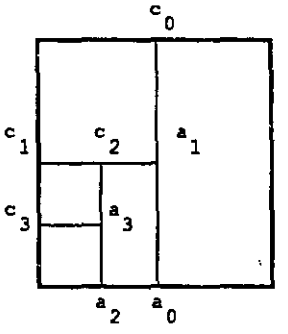


FIGURE 4.1.1.

It is easy to prove that the error in other sub rectangles does not exceed the error examined above.

The algorithm is numerical stable :

4.1.2 LEMMA If ($\forall p : p \in F : n(p) \le \delta$), then

$$|e_n(p)| \le |e_0(p)| \cdot (1 + \frac{\delta}{c})$$

holds for the error e_n .

The proof of this lemma is similar to the previous one using $c + \delta$ instead of c in the formulas (4.1) and (4.2).

4.2 The effect of random noise

Since $E[e_n(p_0)] = (|\pi^*(r',p_0)| + 1)\cdot c$ (where p_0 stands for the same pixels mentioned above), $E[Z(p)] \le \varphi(p) + e_0(p)$ for any $p \in R(A,C)$.

From (4.1) and (4.2) it follows using the same notation as at lemma 4.1.1 that

LEMMA 4.2.3

i) $D^2[Z(p_0)] \le \left(|\pi^*(r',p_0) + \frac{1 - 2^{n+1}}{n+2} \right)\cdot\sigma^2,$

ii) $D^2[Z(a_{2i+1})] \le \frac{1}{2} \left(|\pi^*(r'.a_{2i+1})| + M\cdot 2^i\right)\cdot\sigma,^2$

iii) $D^2[Z(c_{2i})] \le \left(5\cdot 2^{2i-3} + 4\cdot M \left(2^{i-4} + 1 \right) + 4 \right)\cdot\sigma^2.$

From this lemma we can conclude that for any $p \in F$ the magnitude of the error is at most $D[Z(p)] \sim O(M)$ and proportional to the standard variance of the noise, σ.

As we have seen, the error of the digital integration and also the error due to the estimation of the normal vector field depend on $|\pi^*(r',p)|$. This path length can be reduced by setting the reference point to the center of the figure **F**.

4.3 Some experiments of surface reconstruction

Experiments were performed using the quadratic and cubic method for a number of synthetic surfaces. Random noise was introduced to the partial derivatives to simulated uniform and normal distributed of the input. The reference point is chosen in the center of the figure. Algorithm Reconstruction was executed in the following order for the sub rectangles of R(A,C) : R(**r**,C), R((A.x,**r**.y),(**r**.x,C.y)), R(A,**r**) and R((**r**.x,A.y),(C.x,**r**.y)).

The average absolute error between the real surfaces and its reconstruction are shown in table 4.3.1 to compare the results of the the method of Wu and Li, quadratic method and the cubic method with number 1, 2 and 3 respectively.

Surface : sphere, r=12.2, over R((-8,-8),(8,8))[3]						
σ	n is norm. dist., $D(n)=\sigma$			n is uniform dist.in $[-\sigma,\sigma]$		
	1	2	3	1	2	3
0	3.473e-2	7.212e-3	6.129e-4	3.473e-2	7.112e-3	6.129e-4
0.01	3.595e-2	2.219e-2	2.361e-2	-[4]	1.013e-2	9.968e-3
0.1	0.160	0.204	0.236	-	6.810e-2	9.968e-3
1	1.635	2.047	2.368	-	0.672e-1	9.990e-2
Surface : paraboloid, $z = 12.2^2-x^2-y^2$, over R((-16,-16),(16,16))						
0	0	0	0	0	0	0
0.01	1.649e-2	2.248e-2	1.539e-2	-	8.984e-3	1.227e-2
0.1	0.164	0.224	0.153	-	8.984e-2	0.122
1	1.648	2.248	1.539	-	0.898	1.227
Surface : saddle, $z = x^2-y^2$, over R((-16,-16),(16,16))						
0	0	0	0	0	0	0
0.01	1.067e-2	2.248e-2	1.539e-2	-	8.984e-3	1.227e-2
0.1	0.160	0.224	0.153	-	8.984e-2	0.122
1	1.607	2.248	1.539	-	0.898	1.227

TABLE 4.3.1

Obviously, the results are in favor of the new methods without noise or with small noise in the case of the sphere. At the paraboloid and the saddle the average absolute error of the cubic method is less than of the method of Wu and Li.

The result of the cubic method is better then the quadratic method without noise, as it could be expected from the accuracy of the trapezoidal and the Simpson's rule.

[3] The experiments for the same sphere were performed with image size 32 × 32 at Wu and Li.

[4] Wu and Li have studied the noise simulation only in the case of normal distribution.

It can be also seen from the table that the error is approximately linear with the standard variance of the noise according to the theoretical analysis given in section 4.2. Since $D^2[n] = \frac{\sigma^3}{3}$, if n is uniform distributed in $[-\sigma,\sigma]$, the results are better in this case, than in the case of normal distribution with variance σ^2.

From table 4.3.2 we can see that the results are quite satisfactory also in the case of much higher resolution. AAE and MRE stand for the average absolute error and maximal relative error respectively :

Surface : "quarter" sphere, $r^2 = 190000$, over R((0,0),(300,300))								
σ	n is norm. dist., $D[n] = \sigma$				n is uniform dist. in $[\sigma,\sigma]$			
	2		3		2		3	
	AAE	MRE	AAE	MRE	AAE	MRE	AAE	MRE
0	1.544e-4	8.e-5	3.109e-8	3.e-7	1.544e-4	8.e-5	3.109e-8	3.e-5
1	8.845	5.36%	7.493	7.89%	2.968	3.88%	3.634	6.06%

TABLE 4.3.2

The reconstruction of this quarter sphere is illustrated on the following photos, as a "map" of the surface. The upper squares belong to the quadratic method, the lower ones to the cubic method. On the left we can see the reconstructed, on the right the original surface. The interval between the maximum and the minimum of the values of the original and the reconstructed function (with the same method and noise) has been split in 7 intervals of equal length. Each interval has been assigned a color : white, light blue, rose, dark blue, yellow, green and red respectively (see also table 4.3.3) :

Figure	noise	maximum		minimum of meth.	
		2	3	2	3
4.3.1	$n = 0$	435.8	435.8	99.99	99.99
4.3.2	n is norm. dist. with $\sigma = 1$	452.9	460.1	100.0	98.77
4.3.3	n is uniform dist. in $[-\sigma,\sigma]$	435.8	445.3	95.15	100.0

TABLE 4.3.3

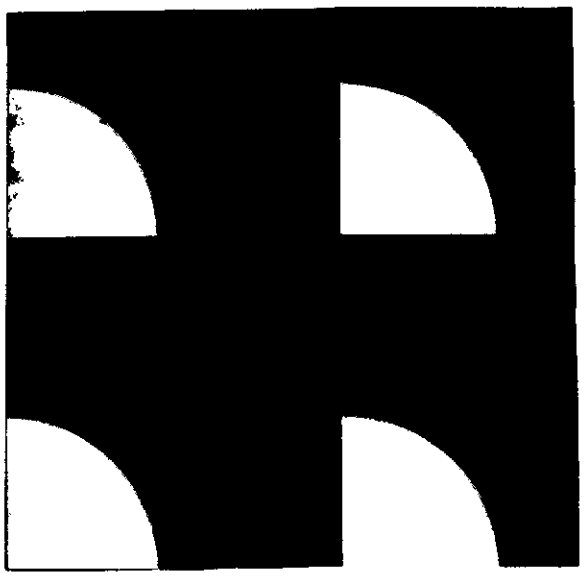


FIGURE 4.3.1

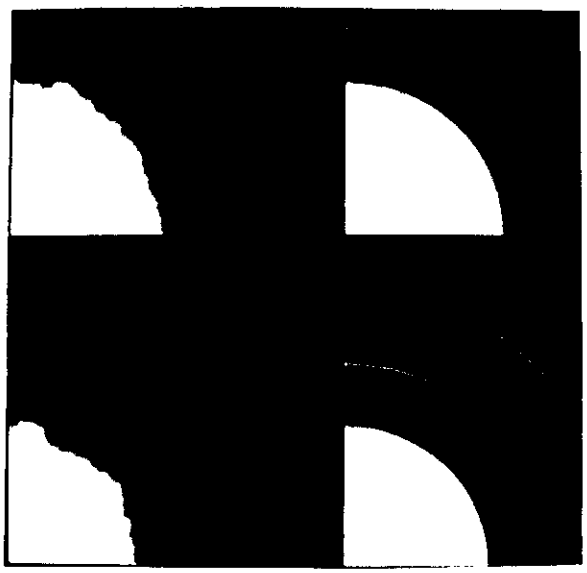
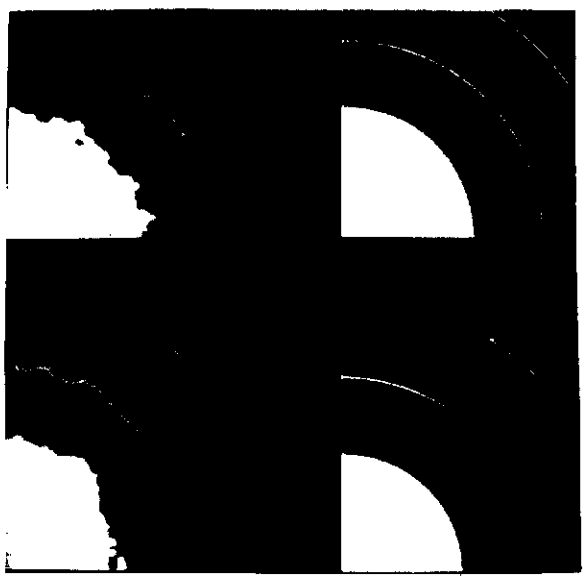


FIGURE 4.3.3

FIGURE 4.3.2



5. CONCLUSION

In this paper a new, general algorithm and its applications are presented to recover depth relative to a reference point of a 3-D surface, from the surface normals. The correctness of this algorithm can be proved using the propositions given in section 2. This method has time complexity $O(|\mathbf{F}|)$ provided that figure **F** covered by distinct rectangles. With resolution M x M the error of the digital integration is $O(M \cdot \log M)$. The algorithm is numerically stable. By choosing the reference point properly, we can reduce the error effects also in the case of error corrupted input. Experiments show that this approach can yield reasonably accurate results.

With respect to step 4) of the reconstruction (see in section 1.2) two generalizations are topics for further investigation :

i) a less homogeneous distribution $f^{a,c}$ in the implementation of procedure Correction (in section 3.1),

ii) figures with arbitrary form.

To segment figures belonging to a continuous normal vector field (step 3) of the reconstruction) we may use the lack of conservativity : large differences in Z value at the same point of the figure, calculated on different path from the reference point may indicate discontinuity in the normal vector field. Orientation discontinuities also can be discovered using decision theory : the change of polynomial grey value distribution can be found with compatibility test based on statistical criteria. From the relation between the grey-value and the surface orientation we may conclude, if discontinuities occur in the normal vector field.

REFERENCES

1. B. K. P. Horn, Understanding image intensities, *Artif. Intel.* **8.**, No. 2, 1977, 201-231.

2. W. M. Silver, Determining Shape and Reflectance Using Multiple Images, Master's thesis, MIT, 1980.

3. B. K. P. Horn and M. J. Brooks, The variational approach to shape from shading, *Comput. Vision, Graphics and Image Process.* **33**, 1985, 173-208.

4. Z. Wu and L. Li, A Line-Integration Based Method for Depth Recovery From surface Normals, *Comput. Vision, Graphics and image Process.* **43**, 1988, 53-66.

5. S. K. Pal and D. K. Dutta, Fuzzy mathematical approach to pattern recognition, Wiley Eastern, New Delhi, 1987.

6. J. Stoer and R. Bulirsch, Introduction to numerical Analysis, Springer-Verlag, Berlin/New York, 1980.

7. V. T. Sós, Analízis I/2. Integrálszámítás, Tankönyvkiadó, Budapest, 1978.

8. E. W. Dijkstra, A Discipline of Programming, Prentice-Hall, Englewood-Cliffs, NJ, 1976.

9. M. v. Lierop, Digitisation Functions in Computer Graphics, Ph.D. Thesis, Eindhoven : University of Technology, 1987.

In this series appeared :

No.	Author(s)	Title
85/01	R.H. Mak	The formal specification and derivation of CMOS-circuits
85/02	W.M.C.J. van Overveld	On arithmetic operations with M-out-of-N-codes
85/03	W.J.M. Lemmens	Use of a computer for evaluation of flow films
85/04	T. Verhoeff H.M.J.L. Schols	Delay insensitive directed trace structures satisfy the foam rubber wrapper postulate
86/01	R. Koymans	Specifying message passing and real-time systems
86/02	G.A. Bussing K.M. van Hee M. Voorhoeve	ELISA, A language for formal specifications of information systems
86/03	Rob Hoogerwoord	Some reflections on the implementation of trace structures
86/04	G.J. Houben J. Paredaens K.M. van Hee	The partition of an information system in several parallel systems
86/05	Jan L.G. Dietz Kees M. van Hee	A framework for the conceptual modeling of discrete dynamic systems
86/06	Tom Verhoeff	Nondeterminism and divergence created by concealment in CSP
86/07	R. Gerth L. Shira	On proving communication closedness of distributed layers
86/08	R. Koymans R.K. Shyamasundar W.P. de Roever R. Gerth S. Arun Kumar	Compositional semantics for real-time distributed computing (Inf.&Control 1987)
86/09	C. Huizing R. Gerth W.P. de Roever	Full abstraction of a real-time denotational semantics for an OCCAM-like language
86/10	J. Hooman	A compositional proof theory for real-time distributed message passing
86/11	W.P. de Roever	Questions to Robin Milner - A responder's commentary (IFIP86)
86/12	A. Boucher R. Gerth	A timed failures model for extended communicating processes

86/13	R. Gerth W.P. de Roever	Proving monitors revisited: a first step towards verifying object oriented systems (Fund. Informatica IX-4)
86/14	R. Koymans	Specifying passing systems requires extending temporal logic
87/01	R. Gerth	On the existence of sound and complete axiomatizations of the monitor concept
87/02	Simon J. Klaver Chris F.M. Verberne	Federatieve Databases
87/03	G.J. Houben J.Paredaens	A formal approach to distributed information systems
87/04	T.Verhoeff	Delay-insensitive codes - An overview
87/05	R.Kuiper	Enforcing non-determinism via linear time temporal logic specification.
87/06	R.Koymans	Temporele logica specificatie van message passing en real-time systemen (in Dutch).
87/07	R.Koymans	Specifying message passing and real-time systems with real-time temporal logic.
87/08	H.M.J.L. Schols	The maximum number of states after projection.
87/09	J. Kalisvaart L.R.A. Kessener W.J.M. Lemmens M.L.P. van Lierop F.J. Peters H.M.M. van de Wetering	Language extensions to study structures for raster graphics.
87/10	T.Verhoeff	Three families of maximally nondeterministic automata.
87/11	P.Lemmens	Eldorado ins and outs. Specifications of a data base management toolkit according to the functional model.
87/12	K.M. van Hee and A.Lapinski	OR and AI approaches to decision support systems.
87/13	J.C.S.P. van der Woude	Playing with patterns, searching for strings.
87/14	J. Hooman	A compositional proof system for an occam-like real-time language

87/15	C. Huizing R. Gerth W.P. de Roever	A compositional semantics for statecharts
87/16	H.M.M. ten Eikelder J.C.F. Wilmont	Normal forms for a class of formulas
87/17	K.M. van Hee G.-J.Houben J.L.G. Dietz	Modelling of discrete dynamic systems framework and examples
87/18	C.W.A.M. van Overveld	An integer algorithm for rendering curved surfaces
87/19	A.J.Seebregts	Optimalisering van file allocatie in gedistribueerde database systemen
87/20	G.J. Houben J. Paredaens	The R^2 -Algebra: An extension of an algebra for nested relations
87/21	R. Gerth M. Codish Y. Lichtenstein E. Shapiro	Fully abstract denotational semantics for concurrent PROLOG
88/01	T. Verhoeff	A Parallel Program That Generates the Möbius Sequence
88/02	K.M. van Hee G.J. Houben L.J. Somers M. Voorhoeve	Executable Specification for Information Systems
88/03	T. Verhoeff	Settling a Question about Pythagorean Triples
88/04	G.J. Houben J.Paredaens D.Tahon	The Nested Relational Algebra: A Tool to handle Structured Information
88/05	K.M. van Hee G.J. Houben L.J. Somers M. Voorhoeve	Executable Specifications for Information Systems
88/06	H.M.J.L. Schols	Notes on Delay-Insensitive Communication
88/07	C. Huizing R. Gerth W.P. de Roever	Modelling Statecharts behaviour in a fully abstract way
88/08	K.M. van Hee G.J. Houben L.J. Somers M. Voorhoeve	A Formal model for System Specification
88/09	A.T.M. Aerts K.M. van Hee	A Tutorial for Data Modelling

88/10	J.C. Ebergen	A Formal Approach to Designing Delay Insensitive Circuits
88/11	G.J. Houben J.Paredaens	A graphical interface formalism: specifying nested relational databases
88/12	A.E. Eiben	Abstract theory of planning
88/13	A. Bijlsma	A unified approach to sequences, bags, and trees
88/14	H.M.M. ten Eikelder R.H. Mak	Language theory of a lambda-calculus with recursive types
88/15	R. Bos C. Hemerik	An introduction to the category theoretic solution of recursive domain equations
88/16	C.Hemerik J.P.Katoen	Bottom-up tree acceptors
88/17	K.M. van Hee G.J. Houben L.J. Somers M. Voorhoeve	Executable specifications for discrete event systems
88/18	K.M. van Hee P.M.P. Rambags	Discrete event systems: concepts and basic results.
88/19	D.K. Hammer K.M. van Hee	Fasering en documentatie in software engineering.
88/20	K.M. van Hee L. Somers M.Voorhoeve	EXSPECT, the functional part.
89/1	E.Zs.Lepoeter-Molnar	Reconstruction of A 3-D surface from its normal vectors.