Journal of Mathematical Imaging and Vision 9, 173–191 (1998)

Computing the Kantorovich Distance for Images

THOMAS KAIJSER
Defence Research Establishment, Division of Command and Control Warfare Technology, Box 1165, S-581 11 Linköping, Sweden
thokai@lin.toa.se

Abstract. Computing the *Kantorovich* distance for images is equivalent to solving a very large transportation problem. The cost-function of this transportation problem depends on which distance-function one uses to measure distances between pixels.

In this paper we present an algorithm, with a computational complexity of roughly order $\mathcal{O}(N^2)$, where N is equal to the number of pixels in the two images, in case the underlying distance-function is the L^1*-metric*, an approximation of the L^2*-metric* or the square of the L^2*-metric*; a standard algorithm would have a computational complexity of order $\mathcal{O}(N^3)$. The algorithm is based on the classical primal-dual algorithm.

The algorithm also gives rise to a *transportation plan* from one image to the other and we also show how this transportation plan can be used for interpolation and possibly also for compression and discrimination.

Keywords: Kantorovich distance, image metrics, Hutchinson metric, transportation problems, primal-dual algorithm

1. Introduction

Roughly speaking the Kantorovich distance for images is defined as the cheapest way to transport one image into the other where the cost is determined by the distance-function chosen to measure distances between pixels.

It was Kantorovich [10, 11] who in the beginning of the 1940s introduced this kind of "transportation-metric" for *probability measures*, and who proved that the metric also can be defined as a supremum of a set of integrals. This result is a special case of what in optimization theory is called the duality theorem.

The first to use the Kantorovich distance as a distance-measure between two-dimensional grey-valued images were probably Werman et al. [17] (see also [18]). The conclusion in their paper was that the Kantorovich distance is applicable in many domains such as co-occurance matrices, shape matching, and picture half-toning. They claim that the match distance, as they call their distance-measure, has many theoretical advantages but also remark that,unfortunately, computing the match distance (Kantorovich distance), is computationally expensive, and that, in some applications the added computation does not result in any substantial improvement. But they also say that when other comparison methods fail, the match distance seems worth considering. In [18] Werman computes the Kantorovich distance in some very simple two-dimensional cases, but otherwise the only computations made are for either one-dimensional images or for images with curves as support.

A few years earlier, in 1981, Hutchinson [7] used the Kantorovich distance for measuring distances between what can be called self-similar probability measures obtained as limiting distributions for a fairly simple type of Markov chains induced by affine, contractive mappings. These limit measures often have supports with *fractal-looking* appearence. Hutchinson used the Kantorovich distance to prove an existence and uniqueness theorem of such limit measures. (This theorem of Hutchinson [7], Section 4.4, Theorem 1, was proved already in the 1930s by Doeblin and Fortet in a substantially more general setting, see [6], Theorem 3.1.)

In the latter part of the 1980s Barnsley and co-workers coined the terminology "iterated function systems" (IFS), and "iterated function systems with probabilities" (IFS with probabilities) for the systems studied by Hutchinson (see e.g., [3, 4]). To prove limit theorems for the case one has an IFS with probabilities, Barnsley and co-workers also use the Kantorovich distance, which they call the Hutchinson metric, but in none of the papers or books of Barnsley and/or co-workers is the Kantorovich distance actually computed.

In 1989 Jacquin published his thesis [8] (see also [9]) in which he describes a new technique to compress images, nowadays often called fractal coding or block-based fractal coding. In his thesis Jacquin also refers to the Kantorovich distance (the Hutchinson metric), but writes that "the main problem with this metric is that it is extremely difficult to compute, theoretically as well as numerically" (see [8], Part I, p. 12).

In probability theory the Kantorovich distance has often been called the Vaserstein metric after the paper [16] by Vaserstein. In [16] Vaserstein defines a *transportation plan* between two probability measures which "puts" as much mass as possible on the diagonal of the product space of the state spaces of the two given probability spaces. A transportation plan between probability measures (stochastic variables, stochastic processes) is nowadays often called a coupling (see, e.g., Lindwall [12]). Important literature on the Kantorovich distance in probability theory are [14] and [15] by Rachev, where also many references to other literature can be found.

As already observed by Werman et al. [17], the main drawback with using the Kantorovich distance is its large computation time. In [17], the authors point out that the use of standard algorithms for transportation problems would imply a computational complexity of order $\mathcal{O}(N^3)$ where N denotes the number of pixels in the two images.

The main purpose of this paper is to present an algorithm for computing the Kantorovich distance function which is *substantially faster* than standard algorithms in case the underlying distance-function between pixels is the L^1*-metric*, the L^∞*-metric*, linear combinations of the L^1*-metric* and the L^∞*-metric*, or the *square* of the L^2*-metric*.

At present our computer programme is such that if we subsample an image, thereby obtaining an image whose side length is $1/2$ of the original image, then the computation time decreases by a factor, $1/\lambda$ say, and in none of the examples we have tried, λ has been larger than $20 \approx 4^{2.2}$.

The factor λ depends on the underlying distance-function as well as on the similarity of the two images and, of course, also on the implementation of the algorithm. Our computer experiments have given λ-values varying from 8 to 20. In case we use the square of the L^2*-metric* as underlying distance-function and the two images are similar then we have found that λ is approximately 16 ($=4^2$). Moreover, in case we have two very dense but non overlapping binary images, with equal total grey value, (the assignment problem), then the factor λ has been as small as 8 ($=4^{1.5}$).

In a recent paper [2], Atkinson and Vaidya have proved that there exists an algorithm which has a computational complexity of order $\mathcal{O}(N^2 \log(N)^3)$ in case one uses the L^1*-metric* as distance-function and of order $\mathcal{O}(N^{2.5} \log(N))$ in case one uses the L^2*-metric* as distance-function. In their paper Atkinson and Vaidya do not give any explicit computation times for their algorithm applied to large transportation problems, and therefore it is not so easy to compare the efficiency of their algorithm with the efficiency of ours. Nor did they apply their algorithm to images.

The underlying method we use to compute the Kantorovich distance is a well-known algorithm called the *primal-dual algorithm*. We have essentially followed the presentation of this algorithm as it is given in the book [13] chapter 12, by Murty. The primal-dual algorithm is also described in the books [5] and [1].

The main reason we have managed to decrease the computational complexity of the primal-dual algorithm is that we are able to compute the so-called admissible arcs in an efficient way.

The plan of this paper is as follows. In the next section we introduce some concepts, in particular, the notion of a *transportation image*, and in Section 3 we give the definition of the Kantorovich distance between images in terms of transportation images. We first give the definition for images with equal total grey value, and then a general definition.

In Section 4 we formulate the Kantorovich distance as a linear programming problem, and in Section 5 we present the dual formulation of this linear programming problem.

In Section 6 we present some computational data when computing the Kantorovich distance between the well-known Lenna image (subsampled to an image of size 256×256) and a fractal coded Lenna image, when the underlying distance-function is either the L^1*-metric* or the square of the L^2*-metric*. The computation times on a SUN4/690 machine (SuperStark) are *approximately* 1 h for these examples. We also

show two graphs representing *optimal transportation images*.

In Section 7 we discuss the possibility of using transportation images for *coding*, and in Section 8 we show how a transportation image between two images can be used for *interpolation*.

In Section 9 we introduce another concept associated with a transportation image, namely the *distortion image*. The distortion image is introduced in order to detect image *discrepancies*.

In Section 10 we make a short digress and define the Kantorovich distance for probability measures, and present the duality theorem in this situation.

In Sections 11 to 21 we describe the algorithm which we have used to compute the Kantorovich distance. In Section 11 we give a general outline of the primal-dual algorithm for the general balanced transportation problem, and then in Sections 12 to 21 we present the details. It is in Sections 19 and 20 we present the *crucial ideas* which have made it possible to compute the Kantorovich distance for images of size 256×256 in, so to speak, finite time.

In Section 22 we present an image of a set of optimal dual variables, and in Section 23 we conclude the scientific part of our paper by formulating a complex but challenging problem related to the work described in the paper. We end our paper with acknowledgments.

2. The Transportation Image

We shall start the definition of the Kantorovich distance with a notion we have chosen to call a transportation image.

A *transportation image* is a set

$$T = \{(i_n, j_n, x_n, y_n, m_n),\ 1 \le n \le N\}$$

of finitely many five-dimensional vectors. If nothing else is said, we assume that the last element in each five-vector of a transportation image is *strictly positive*, and we also assume that there are *never* two vectors in a transportation image for which the first four elements are equal. We call a generic vector in a transportation image, a *transportation vector*, we call the first pair of elements the *transmitting pixel*, we call the second pair of elements the *receiving pixel*, we call the fifth element the *mass* element, and we call a pair $((i, j), (x, y))$ of a transmitting pixel (i, j) and a receiving pixel (x, y) an *arc*.

Before we proceed we need to specify what we mean by an image. Let K be a finite set of integer-valued pairs (i, j). By an image P with support K we mean an *integer-valued* nonnegative function $p(i, j)$ defined on K.

Now given a transportation image, we can define two images—a *transmitting image*, P_1 say, and a *receiving image*, P_2 say, as follows: First, let K_1 denote the union of all transmitting pixels in the transportation image and similarly let K_2 denote the union of all receiving pixels. Next, for $(i, j) \in K_1$, let $A(i, j)$ denote the set of indices in the set $\{(i_n, j_n, x_n, y_n, m_n),\ 1 \le n \le N\}$ for which the transmitting pixel is equal to (i, j) that is $(i_n, j_n) = (i, j)$. Similarly, for $(x, y) \in K_2$, define $B(x, y)$ as the set of indices in the set of transportation vectors for which the receiving pixel is equal to (x, y). We now define the *transmitting image* by

$$P_1 = \left\{ P_1(i, j) = \sum_{n \in A(i,j)} m_n, (i, j) \in K_1 \right\}$$

and similarly we define the *receiving image* by

$$P_2 = \left\{ P_2(x, y) = \sum_{n \in B(x,y)} m_n, (x, y) \in K_2 \right\}.$$

From the way P_1 and P_2 are defined it is clear that the total grey value of the transmitting image and the receiving image are the same namely equal to the sum $\sum_{n=1}^{N} m_n$. If a transformation image T has P as transmitting image and Q as receiving image then we say that T is from P to Q.

3. The Definition

Let $P = \{p(i, j), (i, j) \in K_1\}$ and $Q = \{q(x, y), (x, y) \in K_2\}$ be two given images defined on two sets K_1 and K_2, respectively. K_1 and K_2 may be the same, overlap or be disjoint.

In order to define the Kantorovich distance we need to specify a *distance-function* $d(i, j, x, y)$ from an arbitrary pixel (i, j) in the support K_1 of the image P to an arbitrary pixel (x, y) in the support K_2 of the image Q. This distance-function need not be a metric, but such a choice has an advantage in a sense which we will make precise later.

We shall first give the definition of the Kantorovich distance in case the two images P and Q have *equal total grey value*. Let $\Theta(P, Q)$ denote the set of all transportation images from P to Q. Since we have specified a distance-function we can now define the *cost* $c(T)$ for any transportation image $T = \{(i_n, j_n, x_n,$

$y_n, m_n)$, $1 \le n \le N\}$ from P to Q, simply as

$$c(T) = \sum_{n=1}^{N} d(i_n, j_n, x_n, y_n) \cdot m_n.$$

Finally, we define the Kantorovich distance $d_K(P, Q)$ between P and Q—with respect to the distance-function $d(i, j, x, y)$—by

$$d_K(P, Q) = \inf\{c(T),\ T \in \Theta(P, Q)\}.$$

An *optimal transportation image* T from P to Q satisfies $c(T) = d_K(P, Q)$.

Now suppose that we have two images P and Q for which the *total grey values are different*. Let $L(P) = \sum_{K_1} p(i, j)$ denote the total grey value of P, and let $L(Q) = \sum_{K_2} q(x, y)$ denote the total grey value of Q. Let $G(P, Q)$ denote the largest common divisor of $L(P)$ and $L(Q)$, define $\bar{L}(P) = L(P)/G(P, Q)$ and $\bar{L}(Q) = L(Q)/G(P, Q)$. By using the numbers $\bar{L}(P)$ and $\bar{L}(Q)$ we can now define two new images $\bar{P}$ and $\bar{Q}$, with equal total grey value, by multiplying each pixel value of the image P by $\bar{L}(Q)$, and multiplying each pixel value of Q by $\bar{L}(P)$. We then simply define $d_K(P, Q)$ by

$$d_K(P, Q) = d_K(\bar{P}, \bar{Q})/(\bar{L}(P)\bar{L}(Q)).$$

In the rest of the paper we shall always assume that we have two images with equal total grey value.

4. A Linear Programming Formulation

Another way, and perhaps a more straightforward way to define the Kantorovich distance is as follows. Let again $P = \{p(i, j),\ (i, j) \in K_1\}$ and $Q = \{q(x, y), (x, y) \in K_2\}$ be two given images defined on two sets K_1 and K_2, respectively, and assume that the images have *equal total grey value*.

Let $\Gamma(P, Q)$ denote the set of all nonnegative mappings $m(i, j, x, y)$ from $K_1 \times K_2 \to R^+$ such that

$$\sum_{(x,y)\in K_2} m(i, j, x, y) \le p(i, j), \quad \forall (i, j) \in K_1 \quad (1)$$

and

$$\sum_{(i,j)\in K_1} m(i, j, x, y) \le q(x, y), \quad \forall (x, y) \in K_2. \quad (2)$$

We call any function in $\Gamma(P, Q)$ a *transportation plan* from P to Q. A transportation plan for which we have equality in both (1) and (2) will be called a *complete transportation plan* and we denote the set of all complete transportation plans by $\Lambda(P, Q)$.

It is important to notice that to every transportation plan $m(i, j, x, y) \in \Gamma(P, Q)$ there corresponds a unique transportation image

$$T = \{(i_n, j_n, x_n, y_n, m_n), 1 \le n \le N\}$$

defined simply by

$$T = \{(i, j, x, y, m(i, j, x, y)),\ m(i, j, x, y) > 0\}.$$

Note, however, that T has transmitting image P and receiving image Q only if the given transportation plan is complete.

Conversely, if we are given a transportation image

$$T = \{(i_n, j_n, x_n, y_n, m_n), 1 \le n \le N\}$$

between two images P and Q, then we can find a unique function $m(i, j, x, y) \in \Gamma(P, Q)$ simply by defining

$$m(i_n, j_n, x_n, y_n) = m_n, \quad 1 \le n \le N$$

and defining

$$m(i, j, x, y) = 0$$

elsewhere. Recall that in our definition of a transportation image we assumed that there do not exist two or more transportation vectors with the same values on the first four elements and therefore the function defined above is well-defined.

Now let as above $d(i, j, x, y)$ denote a distance-function between pixels in the set K_1 and the set K_2. The Kantorovich distance $d_K(P, Q)$ can then also be defined by

$$d_K(P, Q) = \inf\left\{ \sum_{i,j,x,y} m(i, j, x, y) \cdot d(i, j, x, y),\ m(\cdot, \cdot, \cdot, \cdot) \in \Lambda(P, Q) \right\}. \quad (3)$$

Since, we require that the function $m(i, j, x, y)$ is nonnegative and the constraints defining the functions in the set $\Lambda(P, Q)$ are linear relations we see that the

definition of the Kantorovich distance is equivalent to the formulation of a *linear programming problem* called the *balanced transportation problem*. We call (3) the primal formulation of the Kantorovich distance.

The reason that one cannot apply standard algorithms directly for computing the Kantorovich distance is that the size of the transportation problem we obtain is quite large. If, for example, we consider two images each of size 256×256, then the number of sources and the number of sinks in our transportation problem are 65536, the number of unknowns (variables) is $2^{32} = 4299801236$ and the number of constraints is $2 \times 65536 = 131072$. The standard estimate for how long time a standard algorithm will take to solve an integer-valued transportation problem with essentially equal numbers of sources and sinks is that the time is proportional to N^3 where N is the number of sources. This implies that in case we have images of size 256×256 then the number of operations would be roughly of order 2×10^{14}.

5. The Dual Formulation

Since the computation of the Kantorovich distance is equivalent to solving a transportation problem, and the method we shall use is based on the so-called primal-dual algorithm, we shall now present the *dual formulation*—the dual version—of the linear programming problem defined by (3).

Thus, let $\{\alpha(i, j), (i, j) \in K_1\}$ and $\{\beta(x, y), (x, y) \in K_2\}$ denote variables associated to the pixels in the sets K_1 and K_2, respectively. We call these variables the *dual variables*. As before, let $d(i, j, x, y)$ denote a distance-function from pixels in K_1 to pixels in K_2, and let Ψ denote all sets of dual variables satisfying

$$d(i, j, x, y) - \alpha(i, j) - \beta(x, y) \geq 0,$$
$$(i, j) \in K_1, (x, y) \in K_2.$$

The dual formulation of the Kantorovich distance is as follows:

$$d_K(P, Q) = \sup\left\{ \sum_{K_1} \alpha(i, j) \cdot p(i, j) + \sum_{K_2} \beta(x, y) \cdot q(x, y), \{\alpha(i, j), \beta(x, y)\} \in \Psi \right\}. \quad (4)$$

That the primal and dual formulations of the Kantorovich distance are equivalent is well known from optimization theory. (For a fairly short proof of this result see, e.g., [1], Appendix C.6, where a proof based on the simplex method is given. Another proof based on graph theory is also given in [1], Section 9.4. See also the paper [10] by Kantorovich.)

Before concluding this section let us also mention that in case $K_1 = K_2 = K$ and the underlying distance-function is a metric, then the dual formulation can also be formulated as follows. First, let Ψ_0 denote all sets $\{\alpha(i, j), (i, j) \in K\}$ such that $\alpha(i, j) - \alpha(x, y) \leq d(i, j, x, y), \forall (i, j), (x, y) \in K$. Then

$$d_K(P, Q) = \sup\left\{ \sum_{(i,j)\in K} \alpha(i, j) \cdot (p(i, j) - q(i, j)), \{\alpha(i, j)\} \in \Psi_0 \right\}. \quad (5)$$

That the definitions (4) and (5) are equivalent, in case the distance-function is a metric, follows from the duality theorem (see Theorem 10.1) and the fact that in case $d(i, j, x, y)$ is a metric then, because of the triangle inequality, the Kantorovich distance between P and Q is equal to the Kantorovich distance between $P^* = \{p^*(i, j)\}$ and $Q^* = \{q^*(i, j)\}$ defined by

$$p^*(i, j) = \max(p(i, j) - q(i, j), 0) \quad (6)$$

and

$$q^*(i, j) = \max(q(i, j) - p(i, j), 0). \quad (7)$$

We shall not begin to describe the algorithm we have used to compute the Kantorovich distance until Section 11. Before that we shall present two examples and indicate some possibilities how to utilize an optimal transportation image.

6. Examples

We let P denote the well-known image of Lenna of size 256×256 (see Fig. 1) and we let Q be an approximation of the Lenna image (also of size 256×256) obtained from a block-based fractal coder (see Fig. 2). (We changed the grey values by one unit in approximately 100 pixels, so that the two images would have the same total grey value).

Figure 1. The original Lenna image subsampled to 256×256.

Figure 2. A 256×256 approximation of the Lenna image obtained by using a fractal block coder based on triangle blocks.

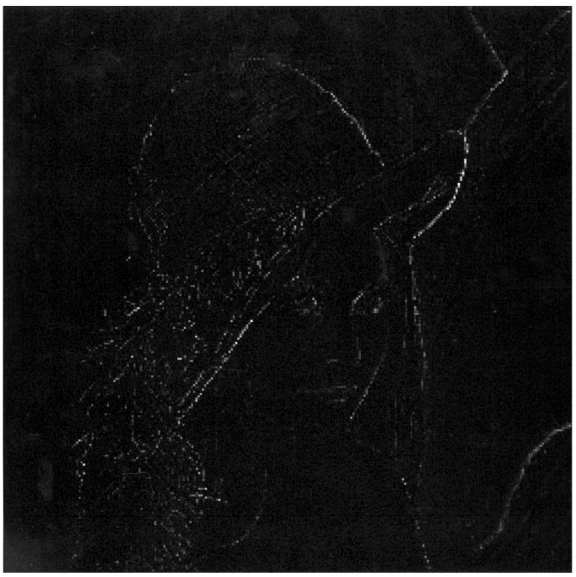

Figure 3. The positive difference between the images in Figs. 1 and 2 (in that order).

Figure 4. The positive difference between the images in Figs. 2 and 1 (in that order).

As underlying distance-function in our first example we have used the L^1-*metric* defined by

$$d(i, j, x, y) = |i - x| + |j - y|.$$

Since we have chosen a metric as underlying distance-function, we can start our computation by first subtracting the common part $R = \{R(i, j), 1 \leq i \leq 256, 1 \leq j \leq 256\}$ defined by

$$R(i, j) = \min(P(i, j), Q(i, j))$$

from both P and Q, thereby obtaining two new images P^* and Q^* defined by (6) and (7) (see Figs. 3 and 4).

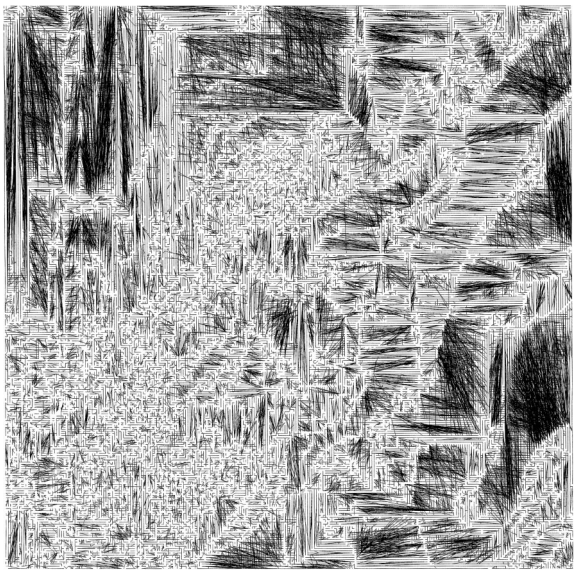

Figure 5. The arcs of positive length of an optimal transportation image from an original Lenna image of size 256×256 to an approximation, when $d(i, j, x, y) = |i - x| + |j - y|$ is used as distance-function.

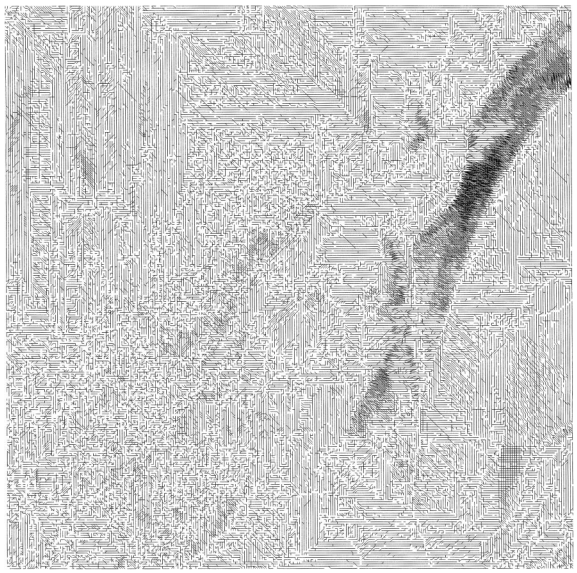

Figure 6. The arcs of positive length of an optimal transportation image from an original Lenna image of size 256×256 to an approximation, when $d(i, j, x, y) = (i - x)^2 + (j - y)^2$ is used as distance-function.

From the images of Figs. 3 and 4 we note that the coded image did not succeed to approximate neither the edges of the mirror nor the feather of the hat of the Lenna image very well.

In our example the total grey value of P^* and Q^* is equal to 232161, the number of nonzero pixel values in P^* and Q^* are 29990 and 30679, respectively, which gives a total of 60669 pixels. The computation time to compute the Kantorovich distance on a Sun4/690 machine, using an implementation of the algorithm by Tech. Lic Niclas Wadströmer, is about 1 h, and the computed value is equal to 1526233. In the Fig. 5 we show the positive arcs of an optimal transportation image from P to Q without cycles. (A cycle is defined in Section 13 and cycles are dicussed in Section 17). The number of arcs in this optimal transportation image is 57014.

Just as is the case with the images depicted in Figs. 3 and 4, also the image in Fig. 5 can be regarded as a kind of difference image, and it should be possible to detect some of the features from the Lenna image, for example, the side of the mirror, and the sides of Lenna's hat.

In our next example we use the same images P and Q as above, but we now choose as underlying distance-function

$$d(i, j, x, y) = (i - x)^2 + (j - y)^2.$$

In this case we cannot start our computation by subtracting the common part R defined above. In our case the common total grey value of P and Q is 6884218. Our computation leads to a solution consisting of 129108 arcs of which 69650 have positive length. For this choice of images it turns out that the computation time is only slightly longer when we use the square of the Euclidean metric than when we use the L^1-*metric*. In Fig. 6 we show the positive arcs of the solution.

If we compare the solutions obtained when we use the L^1-*metric* and the square of the Euclidean metric we find what one would expect, namely that in the second solution there are fewer longer arcs and more arcs along the diagonals. These facts can in part be seen from the graphs of Figs. 5 and 6.

7. Using Transportation Images for Coding

In this Section we shall present an idea how to use a transportation image for improving an image coder.

Suppose first that the underlying distance-function is a metric. Let

$$T = \{(i_n, j_n, x_n, y_n, m_n),\ 1 \le n \le N\}$$

be an *optimal* transportation image from P to Q, with no *cycles* (see Section 13), and such that the arcs are ordered in such a way that if $n > k$ then $d(i_n, j_n, x_n, y_n) \cdot m_n \le d(i_k, j_k, x_k, y_k) \cdot m_k$. Define $N(p)$ for $0 \le p \le 1$ by

$$N(p) = \min\left\{M, \sum_{n=1}^{M} d(i_n, j_n, x_n, y_n) \cdot m_n \ge p \cdot d_K(P, Q)\right\}.$$

Now let us choose $P = P^*$ and $Q = Q^*$ where P^* and Q^* are defined as in Section 6 and let T denote the optimal transportion image presented in Section 6 and obtained when using the L^1-*metric* as underlying distance-function. It then turns out that $N(0.5) = 4337$, $N(0.75) = 12598$, $N(0.9) = 33875$, $N(0.95) = 45081$, $N(0.99) = 54696$ and $N(1.0) = 57014$. Since $N(0.5) = 4337$ and $N(1.0) = 57014$ it follows that for this example, half of the distance between the images P and Q of Section 6 is, so to speak, "carried" by less than 8% of the arcs. Therefore, if we truncate the transformation image at the number $N(p)$ and use this truncated transformation image together with the approximation obtained by the block-based fractal coded image of Lenna (see Fig. 2) we can construct a new approximating image $Q''(p)$ say, whose Kantorovich distance to the original Lenna image is $(1 - p)$ times $d_K(P, Q)$ as follows.

First, for $0 \le p \le 1$ define

$$T(p) = \{(i_n, j_n, x_n, y_n, m_n), 1 \le n \le N(p)\},$$

let $P(p)$ and $Q(p)$ be, respectively, the transmitting image and the receiving image of $T(p)$, and then define

$$Q''(p) = Q - Q(p) + P(p).$$

It is easy to see that the distance between P and $Q''(p)$ is, in fact, $(1 - p)$ times $d_K(P, Q)$.

Similarly, using the image P as "starting image" we can define an approximating image $P''(p)$ say, by

$$P''(p) = P - P(p) + Q(p)$$

and this time it is easy to see that the distance between P and $P''(p)$ is p times $d_K(P, Q)$.

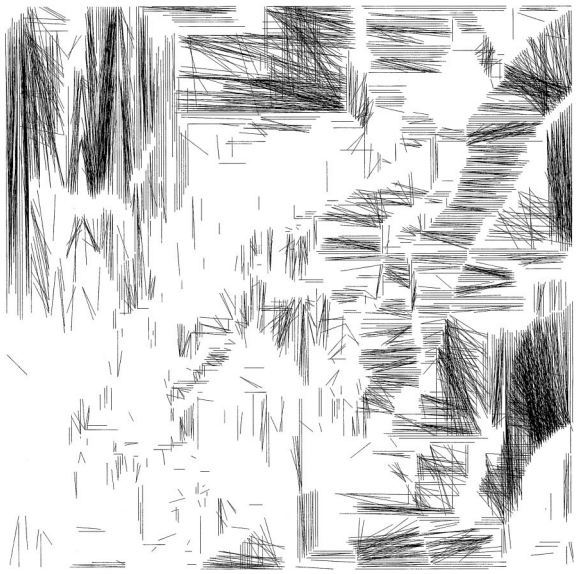

Figure 7. The arcs of a truncation of an optimal transportation image between the images of Figs. 1 and 2, "carrying" 50% of the distance.

In Fig. 7 we have depicted the arcs of the transportation image $T(0.5)$ when T is the solution obtained in the previous section using the L^1-*metric* as underlying distance-function. This part of the original transportation image thus contains most of the longer arcs of the transportation image.

In Fig. 8 we have depicted the image of $Q''(p)$ when $p = 0.5$. This image (as well as the image $P''(0.5)$)

Figure 8. The image is obtained by "adding" a "50% truncated" optimal transportation image to the image in Fig. 2.

has the same Kantorovich distance to both the original Lenna image and the fractal coded approximation.

Although the image of Fig. 8 looks more similar to the original Lenna image then the image of Fig. 2, the improvement is not as good as one would have wished. The reason for this is that most of the longer arcs of the transportation image are allocated to the edges of the mirror and to the edges of the hat in the mirror, and not so many are located to the face of Lenna or to the feather of the hat of Lenna, where much of the discrepancies between the original Lenna image and the fractal coded Lenna image is, as observed by a human eye.

8. Using Transportation Images for Interpolation

There are several ways in which one can use transportation images for interpolation. One way is simply to use the images $Q''(p)$ and $P''(p)$ defined in the previous Section.

Another way is as follows. Let $0 < r < 1$, let

$$T = \{(i_n, j_n, x_n, y_n, m_n),\ 1 \le n \le N\}$$

be a transportation image between the "subtracted" images P^* and Q^* defined by (6) and (7), and define the transportation image

$$T^r = \left\{\left(i_n^r, j_n^r, x_n^r, y_n^r, m_n^r\right),\ 1 \le n \le N\right\}$$

by $i_n^r = i_n$, $j_n^r = j_n$, $x_n^r = i_n + \lfloor(r \cdot (x_n - i_n))$, $y_n^r = j_n + \lfloor(r \cdot (y_n - j_n))$, and $m_n^r = \lfloor(r \cdot m_n)$, where the operation $\lfloor$ means taking the integer part of a number. Let P^r and Q^r denote the transmitting and the receiving image of the transportation image T^r. Finally, define Q^{*r} by

$$Q^{*r} = P - P^r + Q^r.$$

Clearly, as r goes from 0 to 1, the image Q^{*r} goes from the image P to the image Q.

If instead T is a transportation image between the original images P and Q and we redefine T^r slightly, namely by defining $m_n^r = m_n$, then we can use the receiving image of T^r as an interpolating image.

9. Using the Transportation Images for Identifying Discrepancies Between Images

In this section we shall introduce another functional of the transportation image which we call the *distortion matrix*. The idea behind the distortion matrix is that it will be useful when looking for those areas in the two images where the discrepancy between the two images is so to speak the largest. We obtain, in fact, two matrices, one for the pixels in the transmitting image and one for the pixels in the receiving image.

Let $T = \{(i_n, j_n, x_n, y_n, m_n),\ 1 \le n \le N\}$ be a transportation image with transmitting image P and receiving image Q. The distortion matrix $A = \{a(i, j)\}$ say, of the transmitting image P is simply defined as follows:

$$A = \left\{a(i, j) = \sum_n d(i_n, j_n, x_n, y_n) \cdot m_n\right\}$$

where the sum is taken over all vectors in the transportation image which has (i, j) as transmitting pixel, and $d(\cdot, \cdot, \cdot, \cdot)$ as a suitable distance-function. The distortion matrix of the receiving image Q is defined analogously.

In Fig. 9 we have depicted the distortion matrix corresponding to the receiving image Q^* of the optimal transportation image between the images in Figs. 3 and

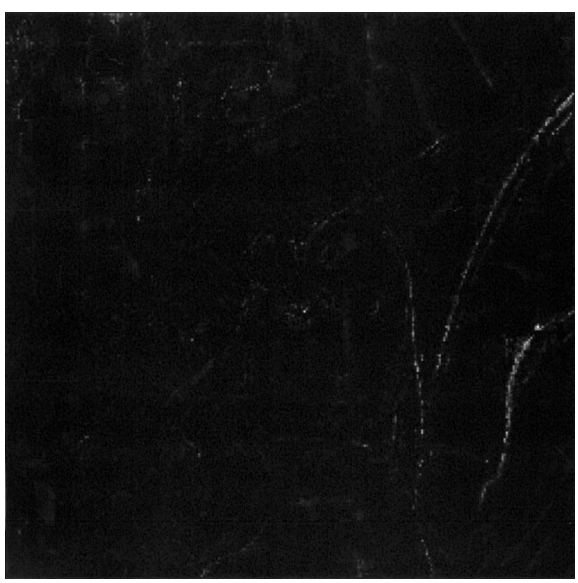

Figure 9. A distortion matrix corresponding to the image in Fig. 4, normalized so that the largest value is 255.

4 presented in Section. In order to be able to present the distortion matrix as an image we have normalized the matrix by first multiplying the distortion matrix by 255 and then dividing by the largest value of the distortion matrix.

When we compare this image with the image Q^* of Fig. 4, we notice that we discover some new discrepancies. For example, we notice that the inner edge of the mirror is more clearly seen, we see more of the upper part of the structure of the wall of the left-hand side of the picture, etc. On the other hand, the feather of Lenna's hat cannot be seen in the image.

10. The Kantorovich Distance for Measures

In this section we shall for sake of completeness present the definition of the Kantorovich distance for positive measures. We shall follow Rachev [14] closely.

Let (U, δ) be a separable metric space, let P_1 and P_2, be two Borel probability measures on (U, δ) and define $\Theta(P_1, P_2)$ as the set of all probability measures P on $U \times U$ with fixed marginals $P_1(\cdot) = P(\cdot \times U)$ and $P_2(\cdot) = P(U \times \cdot)$. Define

$$A_\delta(P_1, P_2) = \inf\left\{ \int_{U\times U} \delta(x, y) P(dx, dy),\ P \in \Theta(P_1, P_2) \right\}.$$

By using the fact that $\delta(x, y)$ satisfies the triangle inequality it is simple to show that $A_\delta(P_1, P_2)$ is a metric.

Next, let

$$\mathrm{Lip}(U) = \{ f : U \to R :\ |f(x) - f(y)| \le \delta(x, y) \}$$

and define

$$B_\delta(P_1, P_2) = \sup\left\{ \left| \int_U f(x) P_1(dx) - \int_U f(x) P_2(dx) \right|,\ f \in \mathrm{Lip}(U) \right\}.$$

That $B_\delta(P_1, P_2)$ also is a metric is also easy to prove. (It is this metric that many people working with fractals and iterated function systems, nowadays call the Hutchinson metric.)

The following duality theorem goes back to Kantorovich.

Theorem 10.1 (Kantorovich [11]). *If U is compact then*

$$A_\delta(P_1, P_2) = B_\delta(P_1, P_2).$$

Duality theorems similar to Theorem 10.1 can also be proved if we in the definition of $A_\delta(P_1, P_2)$ replace the integrand $\delta(x, y)$ by a somewhat more general function, for example, $\delta(x, y)^p$, with $p > 0$. Moreover, if for $p > 1$ we define

$$C_{\delta^p}(P_1, P_2) = \inf\left\{ \left[\int_{U\times U} \delta(x, y)^p P(dx, dy) \right]^{1/p},\ P \in \Theta(P_1, P_2) \right\}$$

then it can be shown that $C_{\delta^p}(P_1, P_2)$ is also a metric. See [14] and references therein.

11. The Primal-Dual Algorithm for the Transportation Problem: A General Outline

In this and subsequent sections we shall describe the algorithm which we have used to compute the Kantorovich distance. We shall use the so-called primal-dual algorithm and will essentially follow the presentation in Murty [13], chapter 12.

In order to simplify notations we shall reformulate our problem in such a way that our problem will be formulated as a general transportation problem.

Let $S_n, 1 \le n \le N$ denote *sources*, $R_m, 1 \le m \le M$ denote *sinks* (destinations), a_n denote the amount of goods at the source $S_n, 1 \le n \le N$, b_m denote the demand of goods at the sinks $R_m, 1 \le m \le M$, $c(n, m),\ 1 \le n \le N,\ \ 1 \le m \le M$ denote the cost to transport one unit of goods from the source S_n to the sink R_m, and finally, $x(n, m)$ denote the amount of goods sent from the source S_n to the sink R_m.

The primal version of the balanced transportation problem is as follows:

$$\text{Minimize} \sum_{n=1}^{N} \sum_{m=1}^{M} c(n, m) \cdot x(n, m) \tag{8}$$

when $x(n, m) \ge 0, 1 \le n \le N,\ 1 \le m \le M$,

$$\sum_{j=1}^{M} x(n, j) = a(n), \quad 1 \le n \le N, \tag{9}$$

$$\sum_{i=1}^{N} x(i, m) = b(m), \quad 1 \le m \le M, \tag{10}$$

and

$$\sum_{n=1}^{N} a(n) = \sum_{m=1}^{M} b(m).$$

Usually, one also assumes that for $1 \le n \le N$ and $1 \le m \le M$,

$$a(n) > 0, \quad b(m) > 0, \quad c(n,m) \ge 0.$$

If we apply this general formulation to the problem of computing the Kantorovich distance (see (3)) then the sources $S_n, 1 \le n \le N$, correspond to the pixels $(i,j) \in K_1$, the sincs $R_m, 1 \le m \le M$, correspond to the pixels $(x,y) \in K_2$, the numbers $a_n, 1 \le n \le N$, correspond to the pixel values $p(i,j), (i,j) \in K_1$, the numbers $b_m, 1 \le m \le M$, correspond to the pixel values $q(x,y), (x,y) \in K_2$, the *cost matrix* $\{c(n,m), 1 \le n \le N, 1 \le m \le M\}$ corresponds to the distance-function $d(i,j,x,y)$ and the variables $x(n,m)$ correspond to the variables $m(i,j,x,y)$.

To each source S_n and each sink R_m we introduce *dual variables* $\alpha(n)$ and $\beta(m)$. If

$$c(n,m) - \alpha(n) - \beta(m) \ge 0, \quad 1 \le n \le N, 1 \le m \le M \tag{11}$$

then we call the set of dual variables *feasible*. We call a pair (n,m) of indices for which n is an index of a source S_n and m is an index of a sink R_m an *arc*. An arc (n,m) such that

$$c(n,m) - \alpha(n) - \beta(m) = 0$$

is called an *admissible arc*. Otherwise, the arc is called *nonadmissible*. By a *flow* we mean any matrix $\{x(n,m), 1 \le n \le N, 1 \le m \le M\}$ of nonnegative elements such that $\sum_{m=1}^{M} x(n,m) \le a(n), 1 \le n \le N$, and $\sum_{n=1}^{N} x(n,m) \le b(m), 1 \le m \le M$. We say that $x(n,m)$ is the flow of the arc (n,m). A flow for which (9) and (10) hold is called an *optimal flow*.

The dual version of the transportation problem is as follows:

$$\text{maximize} \left(\sum_{n=1}^{N} \alpha(n) \cdot a(n) + \sum_{m=1}^{M} \beta(m) \cdot b(m) \right) \tag{12}$$

when the set of dual variables is feasible (satisfies (11)).

That the solution to the primal version is larger than or equal to the solution of the dual version is easily proved by replacing $c(n,m)$ by $(c(n,m) - \alpha(n) - \beta(m)) + \alpha(n) + \beta(m)$ in (8) and then using (9) and (10). In order to prove equality it suffices to have an algorithm which generates both an optimal flow and a feasible set of dual variables such that the flow is zero on any nonadmissible arc. Such an algorithm is the primal-dual algorithm.

Before we can describe the basic steps in the primal-dual algorithm we need a few more concepts. Given a feasible dual solution we call a flow an *admissible* flow if the flow is zero on any nonadmissible arc. Furthermore, if Ψ denotes a set of arcs, we say that the flow *lives on* Ψ, if for any arc not belonging to Ψ, the flow is zero along that arc. We call a flow $\{x(n,m), 1 \le n \le N, 1 \le m \le M\}$ which lives on a set Ψ a *maximal flow* if any other flow $\{y(n,m), 1 \le n \le N, 1 \le m \le M\}$ which lives on Ψ is such that

$$\sum_{m=1}^{M} \sum_{n=1}^{N} y(n,m) \le \sum_{m=1}^{M} \sum_{n=1}^{N} x(n,m).$$

The primal-dual algorithm consists essentially of the following steps:

(0) Find an initial set of dual variables, determine the corresponding initial set of admissible arcs, and find an initial admissible flow.
(1) Check whether the present admissible flow is maximal on the present set of admissible arcs. If it is go to (3). If it is not go to (2).
(2) Update the admissible flow. Then go to (1).
(3) Check whether the present maximal flow is optimal. If it is go to (6). If it is not go to (4).
(4) Update the set of dual variables.
(5) Determine the new set of admissible arcs. Then go to (1).
(6) Ready.

12. The Details of the Primal-Dual Algorithm: The Initialization

We are now ready to go through the algorithm in detail.

To obtain an initial set of feasible dual variables we do exactly as is described in [13], chapter 12. Thus, for $1 \le n \le N$ and $1 \le m \le M$, define

$$\alpha(n) = \min\{c(n,j),\ 1 \le j \le M\},$$
$$\beta(m) = \min\{c(i,m) - \alpha(i), 1 \le i \le N\}.$$

From the definition of $\beta(m)$ it is clear that $\alpha(n) + \beta(m) \leq \alpha(n) + c(n,m) - \alpha(n) = c(n,m)$ and hence (11) is satisfied.

We shall at present not discuss the problem of how to determine the set of admissible arcs, but postpone that until later (Sections 19 and 20).

The simplest choice when defining an initial admissible flow is to define $x_0(n,m) = 0, \forall n, m$. This is what is done in [13] and this is what we also do.

13. Preperations for the Labeling Routine and the Flow-Change Routine

In this section we shall introduce some further useful concepts.

A *path* starting at the source S_i is a sequence $\{(n_l, m_l),\ 1 \leq l \leq L\}$ of admissible arcs, such that (1) $n_1 = i$, (2) if $L > 1$ then $m_{2l-1} = m_{2l}$ and $n_{2l} = n_{2l+1},\ 1 \leq l \leq L/2$, (3) no arc occurs twice, and (4) each source and each sink occurs in at most two arcs. A path starting at a sink R_j is defined analogously. The *length* of the path is equal to the number of arcs in the path.

We say that a path $\{(n_l, m_l),\ 1 \leq l \leq L\}$ goes from the source S_i to the sink R_j if it starts at S_i, the number of arcs is odd and $m_L = j$, and we say that a path goes from the source S_i to the source S_j if it starts at S_i, the number of arcs is even and $n_L = j$.

We say that a source n is *deficient* (with respect to the given flow) if $\sum_{m=1}^{M} x(n,m) < a(n)$. Similarly, we call a sink R_m *deficient* if $\sum_{n=1}^{N} x(n,m) < b(m)$. A source or a sink which is not deficient we call *full*.

An *augmenting path* between a deficient source S_i and a deficient sink R_j is a path $\{(n_l, m_l),\ 1 \leq l \leq L\}$ from S_i to R_j such that if $L > 1$ then

$$x(n_{2l}, m_{2l}) > 0, \quad 1 \leq l < L/2. \tag{13}$$

We end this section introducing two more well-known notions. Let again $\{x(n,m), 1 \leq n \leq N, 1 \leq m \leq M\}$ be an admissible flow, and let $\{(n_l, m_l),\ 1 \leq l \leq L\}$ be a path from a source S_i which returns to S_i. (Thus $n_1 = i, n_L = i$ and L is even.) If also $x(n_l, m_l) > 0,\ 1 \leq l \leq L$ then we say that the path is a *cycle*. A flow without cycles will be called a *forest*.

14. A Trivial Flow-Increasing Step

At point (2) of the general discription of the algorithm described at the end of Section 11, what one has to do is to increase the admissible flow. In this section we shall describe an almost trivial way to increase the flow.

We assume that we have just obtained a new set of admissible arcs. Let A denote the set of all deficient sources, let B denote the set of all deficient sinks, and C denote all *admissible* arcs (i, j) such that the source $S_i \in A$ and the sink $R_j \in B$.

The trivial flow-increasing step consists simply of increasing the flow, as much as possible, along arcs in C as long as there are any arcs left in C. Each time we pick an arc in C and increase the flow along that arc, that arc is excluded from the set C, since either the source or the sink, or both become full. But also other arcs may be excluded after we have increased the flow along an arc (i, j) since either the source S_i or the sink R_j can be endpoints for other arcs belonging to C.

There are many ways one can choose the order for selecting arcs in C. We shall, however, not discuss this issue.

Let us also observe that if we assume that the admissible flow we started with was maximal on the previous set of admissible arcs and also a forest, then the updated flow will also be a forest. The reason for this is that if the new flow value on an arc (i, j) would give rise to a cycle, then the flow we started from could not have been maximal.

At this point we also want to mention that we use the method described above the first time we update the zero flow which we used as our initial flow. It is obvious that if we update the zero flow in this way the updated flow will be a forest.

15. The Flow-Change Routine

In the previous section we increased the flow on augmenting paths of length 1. In this section we shall show how to increase the flow on augmenting paths of length at least 3.

Thus suppose that we have found an augmenting path $\{(n_l, m_l),\ 1 \leq l \leq L\}$ from the source S_i to the sink R_j of length $L \geq 3$. Define θ_1 by

$$\theta_1 = \min\{x(n_{2l}, m_{2l}),\ 1 \leq l < L/2\},$$

a quantity which must be positive because we have an augmenting path (see conditon (13)). We next define θ by

$$\theta = \min\left\{a(i) - \sum_{m=1}^{M} x(i,m),\right.$$
$$\left. b(j) - \sum_{n=1}^{N} x(n,j),\ \theta_1\right\}, \tag{14}$$

a quantity which also must be positive since θ_1 is, and since we have assumed that both the source S_i and the sink R_j are deficient.

We can now obtain a new flow with larger total value, if we redefine the flow values on the arcs of the path as follows:

$$x_{\text{new}}(n_{2l-1}, m_{2l-1}) = x_{\text{old}}(n_{2l-1}, m_{2l-1}) + \theta, \qquad 1 \le l \le (L+1)/2$$

$$x_{\text{new}}(n_{2l}, m_{2l}) = x_{\text{old}}(n_{2l}, m_{2l}) - \theta, \qquad 1 \le l < L/2.$$

It is obvious that we still have a flow and that its total value has increased by θ.

The above procedure to update a flow is called *the flow-change routine*.

Finding augmenting paths and then applying the flow-change routine is thus a way to increase a flow on a given set of admissible arcs. But it is also the only way. For we have:

Proposition 15.1. *Suppose we have a flow on a set of admissible arcs. Suppose also that there is no augmenting path from a deficient source to a deficient sink. Then the present flow is maximal on the present set of admissible arcs.*

The truth of this proposition is intuitively clear. For a formal proof of this result see the pages 177–185 of [1]. Compare also with Kantorovich's proof of the duality theorem [10].

16. The Labeling Routine

The purpose of the labeling routine is to find augmenting paths from deficient sources to deficient sinks. We shall show how this can be done by first describing a labeling routine for a more general setup. We call this the *general labeling procedure*.

Thus, let A and B be two finite sets. We denote a generic element in A by the letter i, and a generic element in B by the letter j. We assume that for each $i \in A$ there is a well-defined set G_i of *neighbors* in B, and for each $j \in B$ there is a well-defined set G_j of neighbors in A, both of which may be empty. Let A_0 and B_0 be subsets of A and B, respectively. The general labeling procedure is a simple procedure to find "all" connections between elements of A_0 and elements of B_0 where we by connection mean that there exists a path from A_0 to B_0 along neighbors, and where we by "all" mean that there does not exist any further element in B_0 which can be reached by a path from an element in A_0.

The general labeling procedure is as follows. Start by labeling all elements in A_0 with a + sign, for example. Then choose any element i_1, say, in A_0 and label all its neighbors in B with the label i_1. Then choose another element i_2, say, in A_0 and label all its neighbors in B *which are not yet labeled* with the label i_2. Continue in this way until all elements in A_0 have been considered. If no element in B is labeled the procedure is completed.

Let B_1 denote those elements in B which have been labeled so far. Each element in B_1 has thus, so to speak, a *parent* in the set A_0. Next, choose an element j_1, say, in B_1, consider its neighbors in A and if there are one or more neighbors which have not yet been labeled, thus do not belong to A_0, label them by j_1. Then choose another element j_2, say, in B_1, consider its neighbors in A and if there are one or more neighbors which have not yet been labeled, label them by j_2. Continue in this way until all elements in B_1 have been considered.

Let A_1 denote those elements in A which have been labeled so far but do not belong to A_0. If A_1 is empty the procedure is completed. Otherwise, choose an element i_1, say, in A_1 and label all its neighbors in B which are not yet labeled with the label i_1. Then choose another element i_2, say, in A_1 and label all its neighbors in B which are not yet labeled with the label i_2. Continue in this way until all elements in A_1 have been considered.

Let B_2 denote those elements in B which have been labeled so far, and which are not in B_1. If B_2 is empty the procedure is completed. Otherwise, continue in exactly the same way as when we had the set B_1, and let A_2 denote those elements in A which have an element in B_2 as parent. If A_2 is empty the procedure is completed. Otherwise, continue the labeling process.

Sooner or later the labeling procedure will end. When that happens we have found *all elements* in B_0 for which there is a path to an element in A_0, and for each such element we can identify a string of parents which leads back to an element in A_0. If an element in B_0 becomes labeled we say that we have a *breakthrough*, otherwise we say that we have a *nonbreakthrough*.

Note that usually the general labeling procedure does *not find all paths* from A_0 to B_0. Note also that the set of paths obtained depends on the *selection order* by which one chooses the elements in A_0, B_1, A_1, B_2, etc.

We shall now apply the general labeling procedure to the more special situation we are considering. As before let A denote the set of sources and B the set of sinks. Let A_0 in the general labeling procedure be the set of deficient sources and let B_0 denote the set of deficient sinks. For an element S_i in A we define its set G_i of neighbors as the set of elements R_j in B for which (i, j) is an admissible arc, and for an element R_j in B we define its set G_j of neighbors as the set of elements S_i in A for which (i, j) is an admissible arc, and for which *also the flow* $x(i, j)$ *is strictly positive*.

Now by applying the general labeling procedure to the situation just specified we obtain a set B^* say, consisting of all deficient sinks for which there exists an augmenting path from some deficient source. If the set B^* is empty, then by Proposition 15.1 the present flow is maximal and we go to the dual solution change routine. Otherwise, for each sink $R_j \in B^*$ we follow the labeling backwards, thereby finding an augmenting path, after which we go to the flow-change routine and update the flow. Since the augmenting paths obtained may have common arcs, it may happen that when we go to the flow-change routine then the value θ defined by (14) may be equal to 0. If this happens we just leave the flow-change routine and go to the next element in B^*.

After we have updated the flow for each sink found in B^*, we start the labeling procedure anew. We continue to go back and forth between the labeling procedure and the flow-change routine until either the labeling procedure results in a nonbreakthrough which implies that the present flow is maximal, or all sources are full in which case we have found an optimal flow.

Our labeling procedure differs slightly from the procedure described in [13], pp. 369, 370. In [13], chapter 12, the labeling process is stopped as soon as the first augmenting path is found.

There are many modifications possible regarding the labeling procedure. For example, instead of considering all deficient sources in parallel, so to speak, one could take one element at a time and develop a "branching tree" for each deficient source. Another possibility is to start from the set of deficient *sinks* every *second* time one applies the general labeling procedure, and whenever one uses the the general labeling anew, one *only* allows sinks and sources which were *labeled* during the *previous* use of the general labeling procedure.

This completes our description of steps (1), (2) and (3) of the primal-dual algorithm as presented in Section 11.

17. How to Avoid Cycles

One drawback with the labeling procedure as it is described in [13] is that, after one have updated the flow using the flow-change routine, one very easily obtains cycles, and our experiments have shown that one often obtains optimal solutions for which the optimal flow may have as much as 15 to 20% more nonzero arcs than necessary.

In order to obtain an optimal flow without cycles, one has several options. One can, for example, construct an algorithm which finds all cycles in a flow, and then redefine the flow along these cycles so that the discovered cycles disappear. Such an "uncycling" procedure one can either use once at the end, when one has found an optimal solution, or one can use it several times during the execution of the primal-dual algorithm, for example, just before one updates the new dual variables.

Another way to avoid obtaining cycles is to apply the general labeling procedure in a slightly different— and somewhat more complicated—way, namely in such a way that one first, as neighboring sets in the general labeling procedure, only allows such sinks as neighbors for which there is a *positive* flow from a source to the sink. By beginning with only "positive" neighbors so to speak, and then slowly but systematicly increasing the sets of neighbors such that, at the end, the neighboring sets of the elements in A contain *all* sinks that can be reached by the admissible arcs, one can make sure that one never obtains any cycles.

18. The Dual Solution Change Routine

There are now only two more parts of the primal-dual algorithm we have to describe, namely step (4) how to compute the new dual variables, and step (5) how to determine the new set of admissible arcs. In this section we consider the problem of how to redefine the dual variables. We assume that we have just applied the labeling procedure to a flow which turned out to be maximal.

Let L_1 denote the set of indices of labeled sources, U_1 denote the set of indices of unlabeled sources, L_2 denote the set of indices of labeled sinks and let U_2 denote the set of indices of unlabeled sinks. Since the flow is maximal but not optimal neither L_1 nor U_2 can be empty.

The dual solution change routine starts by determining the following number:

$$\delta = \min\{c(n, m) - \alpha(n) - \beta(m),\ n \in L_1,\ m \in U_2\}.$$

That δ must be a positive number follows from the definition of L_1 and U_2.

We now change the dual variables as follows:

$$\begin{aligned}
\alpha_{\text{new}}(n) &= \alpha_{\text{old}}(n) + \delta, \quad n \in L_1,\\
\alpha_{\text{new}}(n) &= \alpha_{\text{old}}(n), \quad n \in U_1,\\
\beta_{\text{new}}(m) &= \beta_{\text{old}}(m) - \delta, \quad m \in L_2,\\
\beta_{\text{new}}(m) &= \beta_{\text{old}}(m), \quad m \in U_2.
\end{aligned}$$

It is easy to check that this updated set of variables constitutes a feasible dual solution, that is, we still have $c(n, m) - \alpha(n) - \beta(m) \geq 0, 1 \leq n \leq N, 1 \leq m \leq M$ for the updated set of variables. It is also easy to verify that the old flow, which we knew was a maximal admissible flow on the old set of admissible arcs, is also an admissible flow, but not a maximal flow, on the new set of admissible arcs. (See, e.g., [13], chapter 12 for details.)

The most time-consuming part, in practice, when determining the new dual variables for a general transportion problem, is to determine the number δ. In case one has an integer-valued cost matrix, one can, so to speak, "cheat", simply by *always choosing* $\delta = 1$. When computing the Kantorovich distance for images, this simple choice of δ works quite well for a long time during the execution of the primal-dual algorithm. It is not until one is quite close to the optimal solution that the true value of δ occasionally is larger than 1.

19. Finding admissible Arcs when the Distance-Function is the L^1-*Metric*

The only remaining step to describe in the primal-dual algorithm is how we determine new admissible arcs. In order to describe how this is done, we have to return to our original notations. Thus, we have an image P with support K_1, an image Q with support K_2 and dual variables $\alpha(i, j)$ and $\beta(x, y)$ associated to the pixels in K_1 and K_2, respectively.

As we pointed out when we defined the Kantorovich distance for images, the Kantorovich distance is computed with respect to an underlying distance-function between the pixels of the two images. In this section we shall discuss the case when the underlying distance-function is the L_1-*metric*, that is,

$$d(i, j, x, y) = |i - x| + |j - y|.$$

We first state the following proposition.

Proposition 19.1. *Let the underlying distance-function $d(i, j, x, y)$ be a metric. Let the dual variables $\{\alpha(i, j),\ (i, j) \in K_1\}$ and $\{\beta(x, y), (x, y) \in K_2\}$ be such that for each pixel $(i, j) \in K_1$, there exists a pixel $(x, y) \in K_2$, such that*

$$d(i, j, x, y) - \alpha(i, j) - \beta(x, y) = 0, \tag{15}$$

and similarly that for each pixel $(x, y) \in K_2$, there exists a pixel $(i, j) \in K_1$, such that (15) *holds. Then, if $(i_1, j_1) \in K_1$ and $(i_2, j_2) \in K_1$,*

$$|\alpha(i_1, j_1) - \alpha(i_2, j_2)| \leq d(i_1, j_1, i_2, j_2), \tag{16}$$

and similarly, if $(x_1, y_1) \in K_2$ and also $(x_2, y_2) \in K_2$, then

$$|\beta(x_1, y_1) - \beta(x_2, y_2)| \leq d(x_1, y_1, x_2, y_2). \tag{17}$$

Proof: Let us prove (17). Assume that $\beta(x_1, y_1) - \beta(x_2, y_2) \geq 0$. Let (i_2, j_2) be such that $d(i_2, j_2, x_2, y_2) - \alpha(i_2, j_2) - \beta(x_2, y_2) = 0$. Then

$$\begin{aligned}
&\beta(x_1, y_1) - \beta(x_2, y_2)\\
&\quad = \beta(x_1, y_1) - d(i_2, j_2, x_2, y_2) + \alpha(i_2, j_2)\\
&\quad \leq d(i_2, j_2, x_1, y_1) - \alpha(i_2, j_2) - d(i_2, j_2, x_2, y_2)\\
&\qquad + \alpha(i_2, j_2)\\
&\quad \leq d(x_1, y_1, x_2, y_2)
\end{aligned}$$

where the last inequality sign follows from the triangle inequality. The rest of the proof can be done in an analogous way. □

Before we state and prove the next lemma let us introduce some convenient terminology. Let $(i, j) \in K_1$, let $\alpha(i, j)$ be a dual variable corresponding to (i, j), let $(x, y) \in K_2$ and let $\beta(x, y)$ be a dual variable corresponding to (x, y). If the dual variable $\beta(x, y)$ is such that

$$\beta(x, y) < d(i, j, x, y) - \alpha(i, j)$$

then we say that (x, y) is *low* with respect to (i, j). In case there is little risk for misunderstanding we only say that (x, y) is *low*. In case (x_1, y_1) and (x_2, y_2) are such that

$$d(i, j, x_2, y_2) - \alpha(i, j) - \beta(x_2, y_2) > d(i, j, x_1, y_1) - \alpha(i, j) - \beta(x_1, y_1) > 0$$

then we say that (x_2, y_2) is *strictly lower* than (x_1, y_1).

Let us also introduce the following notations and terminology regarding the positions of two pixels (x_1, y_1) and (x_2, y_2). If $x_1 \leq x_2$ and $y_1 \leq y_2$ we say that (x_2, y_2) is *northeast* (*NE*) of (x_1, y_1). If $x_1 \geq x_2$ and $y_1 \leq y_2$ then we say that (x_2, y_2) is *northwest* (*NW*) of (x_1, y_1). If $x_1 \leq x_2$ and $y_1 \geq y_2$ then we say that (x_2, y_2) is *southeast* (*SE*) of (x_1, y_1). Finally, if $x_1 \geq x_2$ and $y_1 \geq y_2$ then we say that (x_2, y_2) is *southwest* (*SW*) of (x_1, y_1).

The usefulness of the next lemma is that it helps to limit the number of tests needed for finding all new admissible arcs in case we use the L_1-*metric* as underlying distance-function.

Lemma 19.1. *Suppose that the distance-function we are using is the L_1-metric. Let (i, j) be a pixel in K_1, let $\alpha(i, j)$ be a dual variable at (i, j), such that $d(i, j, x_0, y_0) - \alpha(i, j) - \beta(x_0, y_0) = 0$ for some $(x_0, y_0) \in K_2$. Furthermore, assume that for each $(x, y) \in K_2$, there exists some pixel $(i', j') \in K_1$ such that,*

$$d(i', j', x, y) - \alpha(i', j') - \beta(x, y) = 0.$$

Now suppose that $(x_1, y_1) \in K_2$ is low with respect to (i, j). Then

(a) *if (x_1, y_1) is NE of (i, j) and (x, y) is NE of (x_1, y_1) then (x, y) is low,*
(b) *if (x_1, y_1) is NW of (i, j) and (x, y) is NW of (x_1, y_1) then (x, y) is low,*
(c) *if (x_1, y_1) is SE of (i, j) and (x, y) is SE of (x_1, y_1) then (x, y) is low,*
(d) *if (x_1, y_1) is SW of (i, j) and (x, y) is SW of (x_1, y_1) then (x, y) is low.*

Proof: We shall only prove case (a). We prove case (a) by contradiction. Thus suppose there exists a pixel $(x, y) \in K_2$ located NE of (x_1, y_1) and such that at that pixel the dual variable $\beta(x, y)$ is such that $-d(i, j, x, y) + \alpha(i, j) + \beta(x, y) = 0$.

But since (x_1, y_1) is low with respect to (i, j), it follows that $\alpha(i, j)$ must satisfy

$$\alpha(i, j) \leq d(i, j, x_1, y_1) - 1 - \beta(x_1, y_1)$$

which together with the proceeding equality implies that

$$d(i, j, x, y) - \beta(x, y) \leq d(i, j, x_1, y_1) - 1 - \beta(x_1, y_1)$$

and hence $\beta(x, y) - \beta(x_1, y_1) \geq d(i, j, x, y) - d(i, j, x_1, y_1) + 1 = x - i + y - j - (x_1 - i + y_1 - j) + 1 = x - x_1 + y - y_1 + 1 = d(x, y, x_1, y_1) + 1$ which is impossible because of the previous proposition. □

A geometric way to look at this lemma is the following. We know from Proposition 19.1 that $|\beta(x, y) - \beta(u, v)| \leq d(x, y, u, v)$. This means that the graph of the dual variables $\{\beta(x, y), (x, y) \in K_2\}$ looks, so to speak, as a landscape where all slopes are bounded by 1. For fixed (i, j), the distance-function $d(i, j, x, y) = |i - x| + |j - y|$ considered as a function of x and y can be looked upon as an upside-down pyramid. In order to find the admissible arcs having (i, j) as source, what we have to do, so to speak, is to put the top of the pyramid at $(i, j, \alpha(i, j))$ and then find all tangent points to the "surface" $\{\beta(x, y), (x, y) \in K_2\}$. But since the slopes of this "surface" are bounded by 1, as soon as the "surface" is strictly below the pyramid, it will remain to be so, as long as we move away from the center point (i, j).

The lemma above implies that when looking for arcs connected to a labeled pixel (i, j) in K_1, we only have to check pixels (x, y) along a line $y = j_1$ until we have found a pixel (x, y) which is low with respect to (i, j).

20. Finding Admissible Arcs for Other Distance-Functions

In case the underlying distance-function is defined by

$$d(i, j, x, y) = \max\{|i - x|, |j - y|\}$$

(the L^∞-*metric*) a similar stopping rule as given in the previous section can be defined. Also when we let the distance-function be defined as a linear combination of the L^1-*metric* and the L^∞-*metric* (which is a good way to find approximations of the Euclidean metric) it is possible to prove lemmas similar to Lemma 19.1.

However, if the underlying distance-function is the square of the Euclidean distance, that is, if we have

$$d(i, j, x, y) = (i - x)^2 + (j - y)^2 \qquad (18)$$

then it does not seem so easy to prove a lemma analogous to Lemma 19.1.

To define the distance-function by (18) could turn out to be a very useful choice of distance-function, for several reasons. First of all this distance-function is rotationally invariant. Secondly, it turns out that the number of admissible arcs will be substantially smaller than what one obtains when using the L^1-*metric* or an approximate Euclidean metric as underlying distance-function. For the images considered in Section 6, the optimal set of dual solutions gave rise to approximately 3,000,000 admissible arcs in case we used the L^1-*metric* but not more than 257,000 when we used the distance-function defined by (18). Thirdly, if we take the square root, after we have computed the Kantorovich distance, then it turns out that we obtain a *metric*, (see Section 10).

Since the choice of (18) for defining the underlying distance-function is quite attractive, it would, of course, be desirable if a lemma similar to Lemma 19.1 could also be proved in this case. Unfortunately, we have not been able to prove such a lemma.

However, we have find a condition, which, if it holds, implies that the time for the search of new admissible arcs is decreased substantially also when the underlying distance-function is defined by (18). Before we introduce this condition we shall introduce some further terminology concerning the locations of pixels.

Thus, suppose that we have two pixels (i_1, j_1) and (i_2, j_2) belonging to the support K of the same image and such that they are located on the same horizontal line (that is, $j_1 = j_2$). If there is *no* other pixel $(i_3, j_3) \in K$ on the same line as (i_1, j_1) and (i_2, j_2) which is located *between* the pixels (i_1, j_1) and (i_2, j_2), then we say that (i_1, j_1) and (i_2, j_2) are *close* to each other. Furthermore, if $x_2 > x_1$ then we say that (x_2, y_1) is *east* (E) of (x_1, y_1) and if instead $x_2 < x_1$ then we say that (x_2, y_1) is *west* (W) of (x_1, y_1).

Let us now introduce the following assertion.

Assertion 20.1 *Let (i, j) be a pixel in K_1, and let $\alpha(i, j)$ be an admissible dual variable associated to the pixel (i, j) with a value obtained after we have used the "dual solution change" routine.*

Now suppose that (x_1, y_1) and (x_2, y_1) are in K_2 and are close to each other, that both are low with respect to (i, j) and that (x_2, y_1) is strictly lower than (x_1, y_1). Then

(a) *if (x_2, y_1) is E of (x_1, y_1) and (x_1, y_1) is E of (i, j) then all pixels $(x, y_1) \in K_2$ which are E of (x_2, y_1) will also be low, and*
(b) *if (x_2, y_1) is W of (x_1, y_1) and (x_1, y_1) is W of (i, j) then all pixels $(x, y_1) \in K_2$ which are W of (x_2, y_1) will also be low.*

We have not been able to show that Assertion 20.1 always is true, when the distance-function is defined by (18), but so far all our computer experiments have supported it. In case the distance-function is defined by the L^1-*metric* or the L^∞-*metric* the truth of the assertion is trivially true, but also for distance-functions defined as positive integer-valued linear combinations of the L^1-*metric* and the L^∞-*metric* computer experiments have supported the truth of the assertion.

Now, just as Lemma 19.1 makes it possible to construct an algorithm by which we can speed up the search for new admissible arcs when one uses the L^1-*metric*, by assuming that Assertion 20.1 is true, one can introduce a stopping criteria for each line when one is looking for new admissible arcs also when the underlying distance-function is defined by (18).

Hereby, we have completed presentation of our algorithm for computing the Kantorovich-distance for images.

21. Computing the Initial Set of Dual Variables and Admissible Arcs

In Section 12 we described formally how to determine the initial set of dual variables $\{\alpha(n), 1 \le n \le N, \beta(m), 1 \le m \le M\}$. In practice, when we want to compute the initial dual variables, this is not a straightforward task because of the size of the matrix $c(n, m)$. However, to determine the set $\{\alpha(n), 1 \le n \le N\}$ of dual variables, when we have the L^1-*metric*, the L^∞-*metric*, or the square of the Euclidean metric, one can prove that the number of operations is of order N and not NM. Once the dual variables $\alpha(n), 1 \le n \le N$, are determined, one can use Lemma 19.1 or Assertion 20.1 to determine the dual variables $\beta(m)$ as well as the initial set of admissible arcs.

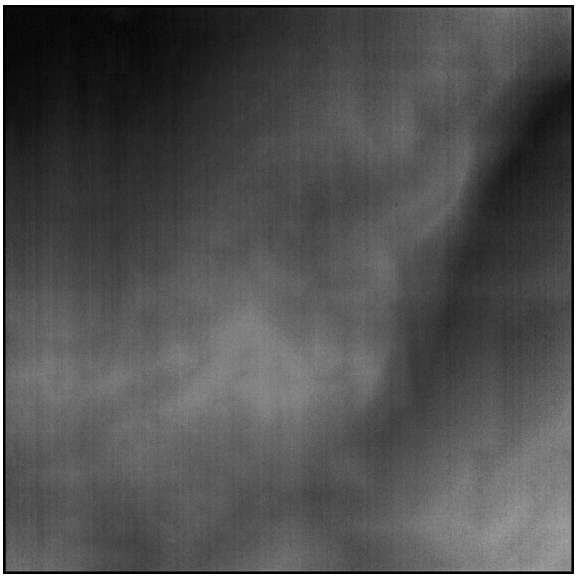

Figure 10. A set of optimal dual variables associated to the pixels of the image of Fig. 1 obtained when computing the Kantorovich distance for the images in Figs. 1 and 2 and when $d(i, j, x, y) = (i-x)^2 + (j-y)^2$.

22. An Image of a Set of Optimal Dual Variables

Let $A = \{\alpha(i, j) : (i, j) \in K_1\}$ and $B = \{\beta(x, y) : (x, y) \in K_2\}$ be the two sets of optimal dual variables obtained for the example considered in Section 6 in case we use the distance-function defined by (18). The purpose of this Section is just to show an image of one of these sets (the dual variables of A). (See Fig. 10).

It is interesting to note that also this image has some slight resemblance of the original Lenna image.

23. A Complex Problem

We shall conclude this paper by describing the following problem.

Let P be a 256×256 image and let Q be a coded version of P with equal total grey value. Let $d(i, j, x, y) = (i-x)^2 + (j-y)^2$, and let A and B denote the two matrices containing the optimal dual variables, obtained when computing the Kantorovich distance between P and Q based on the distance-function $d(i, j, x, y)$. The values of A are associated to the pixels of P and the values of B are associated to the pixels of Q.

Next, let $G(Q)$ denote the set of all images R, say, with equal total grey value as P and Q, such that

$$d_K(R, Q) = R \otimes A + Q \otimes B$$

if we use $d(i, j, x, y) = (i-x)^2 + (j-y)^2$, as distance-function, and where $\otimes$ denotes element-wise multiplication of two matrices. (The set $G(Q)$ is nonempty since $P \in G(Q)$).

Let P^* be the mean of all images belonging to $G(Q)$.

Compute the Kantorovich distance between P and P^*.

Acknowledgments

First of all I want to thank Dr. Robert Forchheimer for both his moral and scientific support during all the years I have been struggling with the computation of the Kantorovich distance. Secondly, I want to thank Dr. Arne Enquist for his patience to listen, and comment on my efforts. Our discussions have been invaluable. Thirdly, I want to thank Tech. Lic Niclas Wadströmer for the image generation programmes and whose skill in programming has been of great value to my work. I want to thank all my other collegues at the Image Coding Group at Linköping University for they always have been very helpful when I have had computer problems. I also want to thank Professor Jan-Olof Eklund for sending me a copy of Werman's thesis [18], and Professor Svante Jansson for pointing out to me the work of Rachev [14]. I want to thank the Swedish National Board of Industrial and Technical Development for financial support.

The main part of this paper was written when I visited New Zealand. I want to thank the Magnusson Foundation for supporting my visit, and want to thank the staff of the Mathematical and Statistical Department of Canterbury University, Christchurch, for their friendliness, hospitality and support during my stay in New Zealand.

References

1. R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows, Theory, Algorithms and Applications*, Prentice Hall: Englewood Cliffs, NJ, 1993.
2. D.S. Atkinson and P.M. Vaidya, "Using geometry to solve the transportation problem in the plane," *Algorithmica*, Vol. 13, pp. 442–461, 1995.

3. M. Barnsley, *Fractals Everywhere*, Academic Press: Boston, MA, 1988.
4. M. Barnsley and L. Hurd, *Fractal Image Compression*, AK Peters: Wellesley, MA, 1993.
5. D.P. Bertsekas, *Linear Network Optimization*, MIT Press: Cambridge, MA, 1991.
6. W. Doeblin and R. Fortet, "Sur des chaines a liaisons completes," *Bull. Soc. Math. France*, Vol. 65, pp. 132–148, 1937.
7. J. Hutchinson, "Fractals and self-similarity," *Indiana J. Math.* Vol. 30, pp. 713–747, 1981.
8. A. Jacquin, "Fractal compression," Ph.D. thesis, Georgia Inst. of Technology: Atalanta, GA, 1989.
9. A. Jacquin, "Fractal coding," *IEEE, Image Compression*, Vol. 1, pp. 49–67, 1992.
10. L. Kantorovich, "On the transfer of masses," (in Russian), *Dokl. Akad., Nauk.*, Vol. 37, No. 2, pp. 227–229, 1942. Translated in *Management Science*, Vol. 5, pp. 1–4, 1959.
11. L. Kantorovich, "On a problem of Monge," (in Russian), *Uspekhi Mat. Nauk.*, Vol. 3, No. 2, pp. 225–226, 1948.
12. T. Lindvall, *Lectures on the Coupling Method*, Wiley: New York, NY, 1992.
13. K. Murty, *Linear and Combinatorial Programming*, Wiley: New York, NY, 1976.
14. S.T. Rachev, "The Monge-Kantorovich mass transference problem and its stochastic applications," *Theory of Prob. Appl.*, Vol. 29, pp. 647–676, 1985.
15. S.T. Rachev, *Probability Metrics*, Wiley: New York, NY, 1990.
16. L. Vaserstein, "Markov processes over denumarable products of spaces describing large systems of automata," *Problems of Information Transmission*, Vol. 5, pp. 64–73, 1969.
17. M. Werman, S. Peleg, and A. Rosenfeld, "A distance metric for multidimensional histograms," *Computer Vision Graphics Image Processing*, Vol. 32, pp. 328–336, 1985.
18. M. Werman, "A distance measure for comparison of pictures and shapes," Ph.D. thesis, The Hebrew University of Jerusalem, Jerusalem, Israel, 1986.

Thomas O. Kaijser received his B.S. degree from Uppsala University in 1965, and the Ph.D. degree in Mathematics from Uppsala University in 1973. From 1973 to 1980 he held a post doctorate research position at Linköping University. Since 1980 he has been employed as applied mathematician at the Defence Research Establishment in Linköping, from 1985 as senior scientist. In 1987 he returned part time to Linköping University as lecturer in mathematics, and since 1990 he has also been associated to the Image Coding Group at Linköping University, where he, in particular, has supported research in fractal image coding.