ON THE COMPUTATION OF THE KANTOROVICH DISTANCE FOR IMAGES

Thomas Kaijser

June 1996

Report nr

LiTH-ISY-R-1858
ON THE COMPUTATION OF THE KANTOROVICH DISTANCE FOR IMAGES

Thomas Kaijser
June 1996

Report nr
LITH-ISY-R-1858
1 Introduction.

The Kantorovich distance for images can be defined in two ways. Either as the infimum of all "costs" for "transporting" one image into the other, where the cost-function is induced by the distance-function chosen for measuring distances between pixels, - or as a supremum of all integrals for which the integration is with respect to the measure obtained by taking the difference of the two images considered as mass distributions, and for which the integrand is any function whose "slope" is bounded. (For the precise definitions see the next sections).

We shall call the first of these two definitions the primal version and the second definition the dual version.

It was Kantorovich [10], (see also [11]), who in the beginning of the 40:ies introduced this "transportation-metric" for probability measures and proved the equality between the two definitions. This result is a special case of what in optimization theory is called the duality theorem.

The first to use the primal version of the Kantorovich distance as a distance-function between 2-dimensional grey valued images with equal total grey value was probably Werman et al [19]. (See also Werman [20]). The conclusions in their paper was that the metric is applicable in many domains such as co-occurrence matrices, shape matching, and picture half-toning. They claim that the metric has many theoretical advantages but also remark that, unfortunately, computing the Kantorovich distance (or rather the match distance, as they call the metric) 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 Kantorovich distance seems worth considering.

In [20] 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 dual formulation of the Kantorovich distance for measuring distances between what can be called self-similar probability measures obtained as limiting distributions for a simple kind of Markov chains induced by affine contractive mappings. Hutchinson used the dual version of the Kantorovich distance to prove an existence and uniqueness theorem of such limit measures, but this theorem ([7], section 4.4, theorem 1) was proved already in the 30:ies by Doeblin and Fortet in a substantially more general setting. (See [6]).

In the latter part of the eighties M.Barnsley and coworkers 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. the books [3] and [4].) They also used the word attractor for the limit measure which always exists for these systems. To prove the existence and the uniqueness of these attractors, and other limit theorems for the case one has an IFS with probabilities, Barnsley and coworkers use the dual version of the Kantorovich metric. In none of the papers or books of Barnsley and/or coworkers is the Kantorovich distance actually computed, and there are very few estimates.

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 or attractor coding. In his thesis Jacquin also refers to the dual version of 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, page 12).

In probability theory the primal version of the Kantorovich distance has often been called the Vaserstein metric after the paper [18] by Vaserstein. In this paper 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 [13]). Couplings have become an important tool in probability theory in particular within the theory of interacting particle systems. (See e.g [12]).

The basic theory on the Kantorovich distance in probability theory, such as the duality theorem in general spaces, has been developed by several researchers among which one should mention Kantorovich, Rubinshtein, Dubroshin, Dudley, Zolotarev, and Rachev in particular. In the paper [16] by Rachev a very general duality theorem is proved (Theorem 3), and also a thorough historical review is given, and in the book [17] by the same author, a large part is devoted to analysing the Kantorovich distance.

The author of this paper got interested in the computational aspects of the Kantorovich distance, for essentially two reasons. The first reason was simply based on the idea that the Kantorovich distance could be useful as a distance-function between images of fractal character when trying to find a solution to the so called inverse IFS-problem. (The inverse IFS-problem means roughly the following: Given an image, find an IFS with as few affine mappings as possible, whose attractor is close to the given image.) If the given image is of fractal character it seems at least possible that the Kantorovich distance could be useful as a tool for discriminating between approximating images and the original image when looking for an answer to the IFS-problem, since the Kantorovich distance utilizes the spacial differences between images and
not only differences in grey values.

The other reason was simply that several authors had claimed that it was more or less impossible to compute the Kantorovich distance because of its high computational complexity, but that it seemed as very few had actually tried to compute the Kantorovich distance for images, even for images with a comparatively small number of pixels.

In [19], Werman et al. point out that the standard algorithm for solving integer valued transportation problems has an estimated computational complexity of order \( O(N^3) \) where \( N \) denotes the number of pixels in the two images. Since this estimate is obtained for transportation problems with very general cost-functions, and since the cost-function when computing the Kantorovich distance has much structure, it seemed at least possible that one would be able to obtain a lower computational complexity than \( O(N^3) \).

It seems now as the order of computational complexity ought to be approximately \( O(N^2) \), which however still is of one power higher than the computational complexity of the ordinary PSNR measure (the \( L^2 \)-metric) which is of order \( O(N) \). At present the computer programme, implemented by the author and based on the algorithm described in this paper, gives a computational complexity of roughly \( O(N^{2.2}) \) in the sense that if we sub-sample 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 many of the examples we have tried, the ratio factor \( \lambda \) is approximately equal to 20 \( \approx 4^{2.2} \), but seldom more. However quite often the ratio factor \( \lambda \) is substantially smaller, sometimes as small as 11. Moreover in case we have two very dense but nonoverlapping binary images, with equal total grey value, (the assignment problem), then the factor \( \lambda \) has been as small as 6.

In the very recent paper [2], Atkinson and Vaidya have proved that there exists an algorithm which is of order \( O(N^{2.5} \log(N))^3 \) in case one uses the Manhattan-metric as distance-function and of order \( O(N^{2.5} \log(N)) \) in case one has the Euclidean metric as distance-function. Their result confirms our initial conjecture that the computational complexity can be lowered thanks to the structure of the cost-function when computing the Kantorovich distance.

We shall make a few further comments on the algorithm of Atkinson and Vaidya in the summary at the end of the paper.

During the author’s work on the computation problems for the Kantorovich distance, the author has become more and more intrigued by the idea to use a transportation plan between images as a starting point for a compression. At first a transportation plan might seem quite costly to code, but we believe that transportation plans can be quite cheap to code, and it is therefore possible that approximations of transportation plans in combinations with other coding methods can yield a quality of the approximated
image which is of the same size or better than what would have been obtained by using the "given" coding technique alone.

Transportation problems are in the modern theory of optimization theory considered as special cases of what is called minimum cost flow problems in network flows. Recently two excellent books have been published on network flows namely the book [5] by Bertsekas and the book [1] by Ahuja, Magnanti and Orlin.

The largest test examples for minimum cost flow problems that they present in their books are 3000 nodes and 15000 arcs, and 8000 nodes and 80000 arcs respectively and the computation times for these examples are 37.38 and 1,064.45 seconds (CPU-time).

Since in principal, the number of nodes is approximately 65,000 and the number of arcs is approximately 1,000,000,000 for the transportation problems that one obtains when computing the Kantorovich distance between images of size 256 x 256, it seems necessary to modify the methods described in their books.

The underlying method we have used to compute the Kantorovich distance has been the so called primal-dual algorithm. We have followed the presentation of this algorithm as it is given in the book [15] chapter 12, by Murty. The primal-dual algorithm is also described in the books [5] and [1].

In principal, we have only modified the primal-dual algorithm in two ways. One step in the algorithm is to determine so called new admissable arcs (new admissable networks) and it is when looking for these arcs that we have managed to reduce the number of pixels one has to investigate substantially.

The other improvement we have made is simply to let the so called labeling process go on until as much labeling has been done as possible. This trick reduces the computation time considerably (approximately by a factor of 3).

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 give the definition of the Kantorovich distance between images in terms of transportation images. In section 3 we formulate the Kantorovich distance as the solution to a linear programming problem, and in section 4 we present the dual version of this linear programming problem.

In sections 5 to 11 we present the primal-dual algorithm for the transportation problem and in section 12 we make some comments regarding the primal-dual algorithm when applied to large transportation problems.

In section 13 we make a short digress and present the Kantorovich distance for probability measures, and present the duality theorem in this situation.

In section 14 we repeat the formulation of the definition of the Kan-
torovich distance for images, in section 15 we briefly discuss how one can handle images with unequal total grey values and in section 16 we describe the so called “subtraction step” which one always can do when the underlying distance-function is a metric.

In sections 17 and 18 we describe the properties that reduce the search for the so called admissible arcs, which decreases the computation time considerably. In section 19 we discuss how to obtain “arc-minimal” solutions.

In section 20 we present som computational data when computing the Kantorovich distance between an approximate image of Lenna and the original Lenna. We also present a figure (Figure 5) showing the arcs of an optimal transportation plan. Otherwise all figures are gathered in an appendix at the end of the paper.

The computation time on a SUN4/690 machine (SuperSparc) is slightly less than one hour for this example.

In section 21 we make some comments about advantages and disadvantages for various possible underlying distance-functions.

In section 22 we discuss the possibility of using transportation plans for coding. In section 23 we introduce the notion of the mean flow vector field of a transportation image and the deviation of the mean flow vector field, both concepts being connected to the problem of approximating a transportation image (transportation plan).

In section 24 we show how a transportation image between two images can be used for interpolation between two images.

In section 25 we introduce another concept associated with a transportation image, namely the distortion image. The distortion image is introduced in order to simplify detection of image discrepancies.

Section 26 finally contains a summary and a discussion.

In Section 27 we make our acknowledgements and in the Appendix we have gathered all our eleven figures.

One final introductionary remark. This paper is written primarily for readers working in image coding and image processing and therefore we have tried to make the paper essentially self-contained. That is why we have been quite elaborate about how the primal-dual algorithm for the transportation problem works.

## 2 The definition.

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

\[ T = \{((i_n, j_n), (x_n, y_n), m_n) : 1 \leq n \leq N\} \]

of finitely many 5-dimensional vectors, where the first two pairs of elements define two pixels (a transmitter and a receiver) and the last element defines a mass "between" the two pixels. If nothing else is said we assume that the last element in each 5-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 \(((i, j), (x, y))\) an arc.

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:

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 let \(A(i, j)\) denote the set of indices in the list

\[ \{((i_n, j_n), (x_n, y_n), m_n) : 1 \leq n \leq N\} \]

defining the transportation image for which the associated transportation vectors are such that their transmitting pixel is equal to \((i, j)\). Similarly define \(B(x, y)\) as the set of indices for which the associated transportation vectors have a receiving pixel equal to \((x, y)\). We now define the transmitting image by

\[ P_1 = \{P_1(i, j) = \sum_{n \in A(i, j)} m_n : (i, j) \in K_1\} \]

and similarly we define the receiving image by

\[ P_2 = \{P_2(x, y) = \sum_{n \in B(x, y)} m_n : (x, y) \in K_2\} \].

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

\[ \sum_{n=1}^{N} m_n. \]

Next let \(P\) and \(Q\) denote two images with equal total grey value. We denote the set of all transportation images which has \(P\) as transmitting image
and $Q$ as receiving image by $\Theta(P, Q)$ or simply by $\Theta$, and we denote a generic element in $\Theta(P, Q)$ by $T(P, Q)$ or simply by $T$. We call any transportation image in $\Theta(P, Q)$ a transportation image from $P$ to $Q$.

Next we need to introduce a distance-function $d(i, j, x, y)$ from a pixel $(i, j)$ in the set $K_1$ to a pixel $(x, y)$ in the set $K_2$. This distance-function need not be a metric, but such a choice has an advantage in a sense which we will make precise later.

Having introduced a distance-function we can now define the cost $c(T)$ for a transportation image $T = \{(i_n, j_n, (x_n, y_n), m_n) : 1 \leq n \leq N\}$ from $P$ to $Q$, simply as

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

where $N$ denotes the total number of transportation vectors in the transportation image under consideration.

Finally the Kantorovich-distance $d_K(P, Q)$ between $P$ and $Q$ - with respect to the distance-function $d(i, j, x, y)$ - is defined as

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

Before finishing this section we shall for sake of completeness introduce a distance for two grey valued images whose total grey values are not necessarily the same.

Thus let $P$ and $Q$ be two given images. Let $L(P)$ and $L(Q)$ denote the total grey value of $P$ and $Q$ respectively. Define $\overline{P} = \{\overline{p}(i, j) : (i, j) \in K_1\}$ and $\overline{Q} = \{\overline{q}(x, y) : (x, y) \in K_2\}$ by

$$\overline{p}(i, j) = p(i, j) / L(P)$$

and

$$\overline{q}(i, j) = q(i, j) / L(Q).$$

Then clearly $\overline{P}$ and $\overline{Q}$ have the same total grey value namely equal to 1. One can therefore define the Kantorovich distance between the two images $\overline{P}$ and $\overline{Q}$ and could then for example define the Kantorovich distance between $P$ and $Q$ simply as

$$d_K(P, Q) = d_K(\overline{P}, \overline{Q}).$$

### 3 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
be two given images defined on two sets $K_1$ and $K_2$ respectively. $K_1$ and $K_2$ might be the same, overlap or be disjoint. We also assume that the images have equal total grey value.

Next let $\Gamma(P, Q)$ denote the set of all non-negative mappings $m(i, j, x, y)$ from $K_1 \times K_2 \rightarrow R^+$ such that

$$\sum_{(x,y)\in K_2} m(i, j, x, y) \leq p(i, j): \ (i, j) \in K_1$$

(1)

and

$$\sum_{(i,j)\in K_1} m(i, j, x, y) \leq q(x, y): \ (x, y) \in K_2.$$  

(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 function $m(i, j, x, y) \in \Gamma(P, Q)$ there corresponds a unique transportation image

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

defined as follows: Let $\Phi(m)$ be defined by

$$\Phi(m) = \{(i, j, x, y): m(i, j, x, y) > 0\},$$

let $N(m)$ be the number of elements in $\Phi(m)$, let $l(n), n = 1, 2, ..., N(m)$ be an “indexing”-function for $\Phi(m)$ (i.e. $l(n)$ is a 1-1 function from the integers $1, 2, ..., N(m)$ to $\Phi(m)$), and finally define the transportation image by

$$T = \{(l(n), m(l(n))): \ 1 \leq n \leq N(m)\}.$$ 

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

Conversely, if we are given a transportation image

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

between two images $P$ defined on the set $K_1$ and $Q$ defined on the set $K_2$, then we can find a unique function $m(i, j, x, y) \in \Gamma(P, Q)$ simply by defining

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

and defining

$$m(i, j, x, y) = 0$$
elsewhere. Recall that in our definition of a transportation image we assumed that there does not exist two or more transportation vectors in a transportation image with the same values on the four first elements and therefore the function defined above is really 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 be defined by

\[
d_K(P,Q) = \min \{ \sum_{i,j,x,y} m(i, j, x, y) \times d(i, j, x, y) : m(\cdot, \cdot, \cdot, \cdot) \in \Lambda(P,Q) \}.
\]

Since we require that the function \( m(i, j, x, y) \) is non-negative 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. In fact, the linear programming problem we obtain is called the balanced transportation problem and there are well-known algorithms for solving such problems.

The reason that one can not 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 \( \times \) 256 then the number of sources and the number of sinks in our transportation problem is 65536, the number of unknowns (variables) is \( 2^{32} = 4299801236 \) and the number of constrains are \( 2 \times 65536 = 131072 \). And if we consider 512 \( \times \) 512 images then the number of sources, sinks and constrains will increase by a factor 4 and the number of unknowns (variables) to be determined will increase by a factor 16.

4 The dual formulation.

In the previous section we saw that the Kantorovich distance can be formulated as the solution of a linear programming problem. Therefore there is also a dual formulation of the Kantorovich distance. 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 of the balanced transportation problem.

The general primal version of the balanced transportation problem can be formulated as follows:

\[
\text{Minimize } \sum_{n=1}^{N} \sum_{m=1}^{M} c(n, m) \times x(n, m)
\]
when

\[ x(n, m) \geq 0, \quad 1 \leq n \leq N, \quad 1 \leq m \leq M, \]

\[ \sum_{j=1}^{M} x(n, j) = a(n), \quad 1 \leq n \leq N, \quad (3) \]

\[ \sum_{i=1}^{N} x(i, m) = b(m), \quad 1 \leq m \leq M, \quad (4) \]

and

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

One also usually assumes that for \( 1 \leq n \leq N, \) and \( 1 \leq m \leq M, \)

\[ a(n) > 0, \quad b(m) > 0, \quad c(n, m) \geq 0. \]

These last three assumptions are usually included as basic assumptions for the transportation problem.

Now let \( \alpha(n), \) \( 1 \leq n \leq N, \) and \( \beta(m), \) \( 1 \leq m \leq M, \) be numbers such that

\[ c(n, m) - \alpha(n) - \beta(m) \geq 0, \quad 1 \leq n \leq N, \quad 1 \leq m \leq M. \quad (5) \]

Let us set

\[ \bar{c}(n, m) = c(n, m) - \alpha(n) - \beta(m). \]

Since

\[ \sum_{n=1}^{N} \sum_{m=1}^{M} c(n, m) \times x(n, m) = \sum_{n=1}^{N} \sum_{m=1}^{M} (\bar{c}(n, m) + \alpha(n) + \beta(m)) \times x(n, m) \]

and since from (3) and (4) it follows that

\[ \sum_{n=1}^{N} \sum_{m=1}^{M} \alpha(n) \times x(n, m) = \sum_{n=1}^{N} \alpha(n) \times a(n) \]

and

\[ \sum_{n=1}^{N} \sum_{m=1}^{M} \beta(m) \times x(n, m) = \sum_{m=1}^{M} \beta(m) \times b(m) \]

we conclude that

\[ \text{minimum} \sum_{n=1}^{N} \sum_{m=1}^{M} c(n, m) \times x(n, m) = \]
minimum \[ \sum_{n=1}^{N} \sum_{m=1}^{M} c(n,m) \times x(n,m) + \sum_{n=1}^{N} a(n) \times a(n) + \sum_{m=1}^{M} \beta(m) \times b(m). \quad (6) \]

Since the first term of the right hand side of this equality must be non-negative if (5) holds, it is clear that the solution of the transportation problem must be greater or equal to the solution of the following linear problem (the dual version of the transportation problem):

\[
\text{maximize} \quad \left( \sum_{n=1}^{N} a(n) \times a(n) + \sum_{m=1}^{M} \beta(m) \times b(m) \right)
\]

when

\[ c(n,m) - a(n) - \beta(m) \geq 0, \quad 1 \leq n \leq N, \quad 1 \leq m \leq M. \]

That in fact the primal and the dual problem have the same solution 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.)

However since a proof of the duality theorem for the transportation problem can be based on the primal-dual algorithm we shall give parts of the necessary arguments which lead to the conclusion of equality.

Let us first suppose that we have a matrix \( \{x(n,m) : 1 \leq n \leq N, 1 \leq m \leq M\} \) which satisfies the constrains (3) and (4) of the transportation problem. Let us also suppose that we have two vectors \( \{a(n), 1 \leq n \leq N\} \) and \( \{\beta(m), 1 \leq m \leq M\} \) satisfying the constrains (5) of the dual problem and also such that whenever \( x(n,m) > 0 \) then

\[ c(n,m) = a(n) + \beta(m). \quad (7) \]

If this is the case we conclude that the first term of the right hand side of (6) is 0, from which we conclude that the matrix \( \{x(n,m) : 1 \leq n \leq N, 1 \leq m \leq M\} \) solves the primal problem and the vectors \( \{a(n), 1 \leq n \leq N\} \) and \( \{\beta(m), 1 \leq m \leq M\} \) solve the dual problem.

Thus in order to prove the equality between the solutions of the primal formulation and the dual formulation of the transportation problem, - and at the same time finding the solution -, it suffices to have an algorithm which produces a matrix \( \{x(n,m) : 1 \leq n \leq N, 1 \leq m \leq M\} \) and two vectors \( \{a(n), 1 \leq n \leq N\} \) and \( \{\beta(m), 1 \leq m \leq M\} \) such that firstly they satisfy the
constrains of the primal and dual problem respectively, and secondly in case 
\( x(n, m) > 0 \) then condition (7) above holds. Such an algorithm is the so
called primal-dual algorithm which we shall describe in the next six sections.

Before we end this section, as we promised in the beginning of this
section, we shall formulate the Kantorovich distance between two images
\( P = \{p(i, j) : (i, j) \in K_1\} \) and \( Q = \{q(x, y) : (x, y) \in K_2\} \) as the solution
of the following dual problem:

\[
\text{maximize } \sum_{(i,j) \in K_1} \alpha(i,j) \times p(i,j) + \sum_{(x,y) \in K_2} \beta(x,y) \times q(x,y)
\]
\[
\text{when } \quad d(i,j, x,y) - \alpha(i,j) - \beta(x,y) \geq 0, \quad (i,j) \in K_1, (x,y) \in K_2.
\]

5 The primal-dual algorithm for the transportat\ion problem. A general outline.

In this and the next five sections we shall describe the primal-dual algorithm
as it is described in Murty [15]. The reason we spend so much time on
explaining the details of this algorithm is of course because it is this algorithm
which we shall use in order to compute the Kantorovich distance for images.

In this section we introduce some terminology and present the general
outline of the algorithm. In the next five sections we shall present the details.

What we want to do is to find a matrix \( \{x(n, m) : 1 \leq n \leq N, 1 \leq m \leq M\} \)
and two vectors \( \{\alpha(n), 1 \leq n \leq N\} \) and \( \{\beta(m), 1 \leq m \leq M\} \) such that firstly
they satisfy the constrains of the primal and dual problems respectively, and
secondly in case \( x(n, m) > 0 \) then condition (7) above holds. It then follows
from what we said above that \( \{x(n, m) : 1 \leq n \leq N, 1 \leq m \leq M\} \) will be
a solution to the primal version of the transportation problem and that the
two vectors \( \{\alpha(n), 1 \leq n \leq N\}, \{\beta(m), 1 \leq m \leq M\} \) will be a solution to the
dual problem.

Let us begin by introducing some terminology. We call the two vectors
\( \{\alpha(n), 1 \leq n \leq N\} \) and \( \{\beta(m), 1 \leq m \leq M\} \) a dual feasible solution if the
elements of the vectors satisfy

\[
c(n,m) - \alpha(n) - \beta(m) \geq 0, \quad 1 \leq n \leq N, 1 \leq m \leq M.
\]

The elements are then called the dual variables. We shall call an index
associated to an element of the vector \( \{\alpha(n), 1 \leq n \leq N\} \) a source, an index
associated to an element of the vector \( \{\beta(m), 1 \leq m \leq M\} \) a sink, and a pair
of a source and a sink we shall call an arc. (In [15] an arc is called a cell, but the term arc seems to be the most common term used nowadays. Another term used is edge.)

Now given a dual feasible solution \( \{\alpha(n), 1 \leq n \leq N\} \) and \( \{\beta(m), 1 \leq m \leq M\} \) we can define the set of admissable arcs as those pairs of indices \((n, m)\) (arcs) for which
\[
c(n, m) = \alpha(n) + \beta(m).
\]
A pair \((n, m)\) for which this equality does not hold is called inadmissable.

By a flow we mean any matrix \( \{x(n, m) : 1 \leq n \leq N, 1 \leq m \leq M\} \) of non-negative elements such that
\[
\sum_{m=1}^{M} x(n, m) \leq a(n), \quad 1 \leq n \leq N \quad \sum_{n=1}^{N} x(n, m) \leq b(m), \quad 1 \leq m \leq M,
\]
and we say that \(x(n, m)\) is the flow of the arc \((n, m)\).

Given a feasible dual solution \( \{\alpha(n), 1 \leq n \leq N\} \) and \( \{\beta(m), 1 \leq m \leq M\} \), we call a matrix \( \{x(n, m) : 1 \leq n \leq N, 1 \leq m \leq M\} \) of non-negative elements, an admissable flow (with respect to the dual solution) if \(x(n, m) = 0\) in case \((n, m)\) is not an admissable arc. If \(\Psi\) denotes the set of admissable arcs we also say that the flow lives on \(\Psi\).

We call a flow \( \{x(n, m) : 1 \leq n \leq N, 1 \leq m \leq M\} \) which lives on a set \(\Psi\) a maximal flow if any other flow \( \{y(n, m) : 1 \leq n \leq N, 1 \leq m \leq M\} \) which lives on \(\Psi\) is such that
\[
\sum_{m=1}^{M} \sum_{n=1}^{N} y(n, m) \leq \sum_{m=1}^{M} \sum_{n=1}^{N} x(n, m).
\]
Finally we call an admissable flow \( \{x(n, m) : 1 \leq n \leq N, 1 \leq m \leq M\} \) optimal if the elements of the matrix satisfy the equalities (3) and (4).

The primal-dual algorithm consists essentially of the following steps:
0) Find an initial feasible dual solution, an initial set of admissable arcs and an initial flow. Then go to 3).
1) Update the set of dual variables. Then go to 2).
2) Determine the new set of admissable arcs. Then go to 3).
3) Check whether the present flow is maximal. If it is go to 5). If it is not go to 4).
4) Update the present flow. Then go to 3).
5) Check whether the present maximal flow is optimal. If it is go to 6). If it is not go to 1).
6) Ready.
6 The details of the primal-dual algorithm.  
The initialization.

We are now ready to go through the algorithm in detail. In this section we shall present how we obtain the initial feasible dual solution, and the initial flow.

To obtain an initial feasible dual solution we do exactly as is described in [15], chapter 12. Thus, first define $c^*(n)$ by

$$c^*(n) = \min \{c(n,m) : 1 \leq m \leq M\}.$$  

Next define our first set of dual variables by

$$\alpha(n) = c^*(n), \quad 1 \leq n \leq N,$$

and

$$\beta(m) = \min \{c(n,m) - \alpha(n) : 1 \leq n \leq N\}, \quad 1 \leq m \leq M.$$  

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 (8) is satisfied and therefore the variables do indeed constitute a set of feasible dual variables.

Next let us denote the set of admissible arcs defined by the initial set of dual variables by $S$. From the definition of the initial set of dual variables it is clear that $S$ consists of at least $N$ arcs, since for each source $n$, there exists at least one sink $m$ for which $c(n,m) = \alpha(n) + \beta(m)$.

We shall next define an initial admissible flow. Let us however first introduce the matrix $\{c(n, m) : 1 \leq n \leq N, 1 \leq m \leq M\}$ by

$$\bar{c}(n, m) = c(n, m) - \alpha(n) - \beta(m).$$

From the definition of $\alpha(n), 1 \leq n \leq N$, and $\beta(m), 1 \leq m \leq M$ it is clear that the matrix

$$\{\bar{c}(n, m) : 1 \leq n \leq N, 1 \leq m \leq M\}$$

has at least one zero in each row and each column. The indices for which the elements of this matrix are zero are precisely the admissible arcs.

There are many ways one can define an initial admissible flow

$$\{x_0(n, m) : 1 \leq n \leq N, 1 \leq m \leq M\}.$$  

The simplest is to define $x_0(n, m) = 0$, $1 \leq n \leq N, 1 \leq m \leq M$, and this is what is done in [15].

Another procedure is as follows. Let $S(1)$ denote the set of indices $m, 1 \leq m \leq M$ for which $(1, m)$ is an admissible arc. Define $y(1)$ by
\[ y(1) = \max\{b(m) : m \in S(1)\} \]
define \( m(1) \) as \( \min\{m : b(m) = y(1)\} \) and define \( x_0(1, m(1)) = \min(a(1), y(1)) \). Next define \( x_0(1, m) = 0 \), for \( 1 \leq m \leq M, \ m \neq m(1) \). Hereby the first row of the matrix defining the initial admissable flow is defined.

Before we define the elements of the second row of the flow matrix let us define \( b(m), 1 \leq m \leq M \), by \( b(m(1)) = b(m(1)) - x_0(1, m(1)) \) and \( b(m) = b(m) \) for \( 1 \leq m \leq M, \ m \neq m(1) \).

Next let us define \( S(2) \) as the set of indices \( m, 1 \leq m \leq M \) for which \((2, m)\) is an admissable arc. Define \( y(2) \) by \( y(2) = \max\{b(m) : m \in S(2)\} \), define \( m(2) \) as \( \min\{m : b(m) = y(2)\} \) and define \( x_0(2, m(2)) = \min(a(2), y(2)) \). Next define \( x_0(2, m) = 0 \), \( 1 \leq m \leq M, \ m \neq m(2) \). Hereby the second row of the matrix defining the initial admissable flow is defined.

And so on. Thus suppose we have defined the \( n\)th row of the initial admissable flow matrix. Let us first update the vector \( b(m), 1 \leq m \leq M \), by \( b(m(n))_{\text{new}} = b(m(n))_{\text{old}} - x_0(1, m(n)) \) and \( b(m)_{\text{new}} = b(m)_{\text{old}} \), \( 1 \leq m \leq M, \ m \neq m(n) \). Next let us define \( S(n+1) \) as the set of indices \( m, 1 \leq m \leq M \) for which \((n+1, m)\) is an admissable arc. Define \( y(n+1) \) by \( y(n+1) = \max\{b(m) : m \in S(n+1)\} \), define \( m(n+1) \) as \( \min\{m : b(m) = y(n+1)\} \) and define \( x_0(n+1, m(n+1)) = \min(a(n+1), y(n+1)) \). Next define \( x_0(n+1, m) = 0 \), for \( 1 \leq m \leq M, \ m \neq m(n+1) \). Hereby the \((n+1)\)th row of the matrix defining the initial admissable flow is defined. And proceeding in this way all rows of the initial admissable flow matrix can be defined.

7 Preperations for the labeling routine and the flow change routine.

The next step in the algorithm is to perform the so called labeling. However before we do this we shall introduce some further terminology and concepts which will make it easier to understand the purpose of the forthcoming labeling routine.

We assume now that we have a feasible dual solution, a set of admissable arcs which is non-empty, and an admissable flow.

A path is a sequence \( \{(n_l, m_l) : 1 \leq l \leq L\} \) of admissable arcs, such that if \( L > 1 \) then \( m_{2l-1} = m_{2l} \), \( 1 \leq l \leq L/2 \) and \( n_{2l} = n_{2l+1}, \ 1 \leq l \leq L/2 \), and such that no arc occurs twice and such that each source and each sink occurs in at most two arcs.

The length of the path is equal to the number of arcs in the path.

Given a path \( \{(n_l, m_l) : 1 \leq l \leq L\} \) we say: 1) if \( L \) is odd, that the path goes from the source \( n_1 \) to the sink \( m_L \), and 2) in case \( L \) is even, that the
path goes from \( n_1 \) to \( n_L \). Furthermore, we say that a source \( n \) and a sink \( m \) are connected if there exists a path going from \( n \) to \( m \); we say that two sources \( n_1 \) and \( n_2 \) are connected if there exists a sink \( m \) such that \( n_1 \) and \( m \) are connected and also \( n_2 \) and \( m \) are connected. Similarly we say that two sinks \( m_1 \) and \( m_2 \) are connected if there exists a source \( n \) such that \( n \) and \( m_1 \) are connected and also \( n \) and \( m_2 \) are connected.

A subset of the set of admissable arcs such that each source in any arc of the subset is connected with each sink of all the arcs in the subset but with no sink not in the subset of admissable arcs is called a connected subset of admissable arcs. Clearly the notion of connectivity constitutes an equivalence relation. We can therefore split the set of admissable arcs into disjoint connected subsets of admissable arcs. From this observation we see that in order to create a maximal flow on a given set of admissable arcs, we could of course begin by splitting the set of admissable cells into disjoint subsets of connected arcs.

Next let \( \{x(n,m) : 1 \leq n \leq N, 1 \leq m \leq M\} \) be an admissable flow. 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 \( m \) deficient if \( \sum_{n=1}^{N} x(n,m) < b(m) \). A source or a sink which is not deficient we call full. By the flow value \( \bar{x}(n) \) of a source \( n \) we simply mean the sum \( \sum_{m=1}^{M} x(n,m) \) and similarly the flow value \( \bar{x}(m) \) of a sink \( m \) is defined as the sum \( \sum_{n=1}^{N} x(n,m) \).

Now in case the present admissable flow is not maximal on this set of admissable arcs, what we want to do is to find an algorithm - a procedure - by which we can change the flow to another admissable flow so that the total value \( \sum_{m=1}^{M} \sum_{n=1}^{N} x(n,m) \) increases.

In order to describe such a procedure we shall need the notion of a good path between a source and a sink defined as follows. We say that there is a good path between a source \( i \) and a sink \( j \) if there exists a path \( \{(n_l, m_l), 1 \leq l \leq L\} \) from \( i \) to \( j \) such that if \( L > 1 \) then

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

A good path between a deficient source and a deficient sink is often called an augmenting path, but we have chosen not to use this terminology.

### 8 The flow-change routine.

Now suppose that we have found a good path between a deficient source \( i \) and a deficient sink \( j \). It is then an easy matter to increase the total value of the flow.
First assume that the length of the path is equal to 1. Then if we define \( \theta \) by

$$
\theta = \min \{ a(i) - \sum_{m=1}^{M} x(i, m), \ b(j) - \sum_{n=1}^{N} x(n, j) \}
$$

it is clear that by redefining the present flow for the index \((i, j)\) by

$$
X_{new}(i, j) = X_{old}(i, j) + \theta,
$$

that we have obtained a flow with larger total flow than before. Moreover either the source \( i \) or the sink \( j \) or both become full.

If instead \( L > 1 \), (that is \( L \) is any positive odd integer \( > 1 \), ) then we define \( \theta_1 \) by

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

a quantity which must be positive because we have a good path. We next define \( \theta \) by

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

a quantity which also must be positive since \( \theta_1 \) is, as we just observed, and we also have assumed that both the source \( i \) and the sink \( j \) 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:

$$
X_{new}(n_{2l-1}, m_{2l-1}) = X_{old}(n_{2l-1}, m_{2l-1}) + \theta, \quad 1 \leq l \leq (L + 1)/2
$$

$$
X_{new}(n_{2l}, m_{2l+1}) = X_{old}(n_{2l}, m_{2l+1}) - \theta, \quad 1 \leq l < L/2.
$$

From the definition of the new flow and the definition of \( \theta \), it is clear that values of all the changed elements still are greater than or equal to zero. Moreover, if we consider the flow values for the sinks and sources along the path from \( i \) to \( j \), (\( i \) and \( j \) excluded) it is clear that the flow values of these do not change, and hence the new flow does indeed satisfy the conditions which constitutes a flow. That the total value of the new flow has increased by \( \theta \) is also clear from the way the new flow is defined.

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

Now, when we redefine the flow on the present set of admissible arcs in this way, either a deficient source or a deficient sink or both a deficient source and a deficient sink become full, or the number of good paths from a deficient source to a deficient sink decrease by at least one. Therefore if we start with a given flow on a set of admissible arcs and update the flow iteratively as long as we can find a good path from a deficient source to a deficient sink
this updating procedure must end after a finite number of iterations since the number of possible good paths are finite and also the number of sources and sinks are finite. Moreover by a little bit of thinking it is not too difficult to convince oneself that the following proposition is true (see also [1], Theorem 6.4):

**Proposition 8.1** Suppose we have a flow on a set of admissible arcs. Suppose also that there is no good path from a deficient source to a deficient sink. Then the present flow is maximal on the present set of admissible arcs.

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].

9 The labeling routine.

The main purpose of the labeling routine is to find a good path from a deficient source to a deficient sink. When that happens we can go to the flow change routine and update the present flow. If it never happens we go to the “dual solution change routine”, which we shall describe in the next section.

The labeling routine is as follows (see [15], page 369):

Let \( \{x(n, m) : 1 \leq n \leq N, 1 \leq m \leq M\} \) be the present admissible flow.

The first part of the labeling routine is to label each deficient source with the label \((s, +)\). If no source is labeled, all sources are full, and we have an optimal flow. If some source is labeled we continue by labeling those sinks by the label \((\text{source } n, +)\) for which the following condition holds:

(i) sink \(m\) is not yet labeled, source \(n\) is labeled, and \((n, m)\) is an admissible arc.

If no sinks satisfy condition (i) we go to the dual solution change routine.

If a deficient sink is labeled then we have found a good path (of length 1) from a deficient source to a deficient sink and we can stop the labeling routine and go to the flow change routine. Finding a good path from a deficient source to a deficient sink is also called obtaining a breakthrough.

If we do not find a good path from a deficient source to a deficient sink the first time we “go through” condition (i) for all labeled sources, but do obtain some labeled sinks, then we continue our labeling by checking the following condition for all labeled sinks:

(ii) source \(n\) is not yet labeled, sink \(m\) is labeled, \((n, m)\) is an admissible arc, and \(x(n, m) > 0\).

If condition (ii) holds we label source \(n\) by \((\text{sink } m, -)\).
If we do not obtain any new labeled rows, we terminate the labeling routine and go to the dual solution change routine. Otherwise we continue by checking condition (i) again, this time for all “newly labeled” sources.

If no so far unlabeled sinks satisfy condition (i), we terminate the labeling routine and go to the dual solution change routine.

If (i) is satisfied for some new sink \( m \), just as before, we label that sink by \((\text{source } n, +)\). If we happen to label a deficient sink we again have a breakthrough, which implies that we have obtained a good path from a deficient source to a deficient sink. The way we find the good path is simply to follow the labeling backwards so to speak, until we come to a deficient source (which is labeled \((s, +)\)). If no new sinks are labeled we terminate the labeling routine and go to the dual solution change routine. Otherwise we check condition (ii) again for those labeled sinks which have not been checked.

And in this way we continue the labeling “going through” conditions (i) and (ii) as long as it is possible. Sooner or later the labeling process will end, either by a so called breakthrough, which means that we can find a good path from a deficient source to a deficient sink, and then we go to the flow change routine, or the labeling routine terminates without obtaining a breakthrough in which case we go to the dual solution change routine.

If we go to the flow-change routine then before we return to the labeling routine we “unlabel” all sinks and sources, and after we have updated the flow by using the flow-change routine we return to the labeling routine and start our labeling afresh. If however we go to the dual solution change routine then we can keep all our labeling and start labeling after we have updated the dual variables and the set of admissible arcs from where it was left off at the occurrence of the nonbreakthrough (to quote Murty [15] verbatim).

10 The dual solution change routine.

We shall now describe the dual solution change routine. The dual solution change routine consists essentially of two parts. The first part consists of determining the new set of dual variables, the second part consists of determining the new set of admissible arcs.

Let \( L_1 \) denote the set of labeled sources, let \( U_1 \) denote the set of unlabeled sources, let \( L_2 \) denote the set of labeled sinks and \( U_2 \) denote the set of unlabeled sinks. Obviously the set \( L_1 \) can not be empty since if this is the case all sources are full and we would have found an optimal flow. Moreover the set \( U_2 \) must also be non-empty because if \( U_2 \) is empty this means that all sinks would be labeled and since at least one sink is not full (since we do not have an optimal flow) the fact that all sinks are labeled would imply that we
would have a good path from a deficient source to a deficient sink, which we cannot have.

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

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

$\delta$ must be a positive number since if the source $n$ is a labeled source and $(n, m)$ is an admissible arc then the labeling procedure would label the sink $m$ and hence $m$ could not then belong to the set $U_2$.

We now change the dual variables as follows:

$$\alpha_{\text{new}}(n) = \alpha_{\text{old}}(n) + \delta, \; n \in L_1$$

$$\alpha_{\text{new}}(n) = \alpha_{\text{old}}(n), \; n \in U_1$$

$$\beta_{\text{new}}(n) = \beta_{\text{old}}(n) - \delta, \; n \in L_2$$

$$\beta_{\text{new}}(n) = \beta_{\text{old}}(n), \; n \in U_2.$$ 

This completes the first part of the dual-solution change routine.

It is easy to check that this updated set of variables constitutes a feasible dual solution, that is that 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. For if $n \in L_1$ and $m \in L_2$ then the sum of $\alpha(n)$ and $\beta(m)$ is unchanged, if $n \in L_1$ and $m \in U_2$ then by the definition of $\delta$, $c(n, m) - \alpha(n) - \beta(m) \geq 0$, if $n \in U_1$ and $m \in L_2$ then the sum of $\alpha(n)$ and $\beta(m)$ decreases, and finally if $n \in U_1$ and $m \in U_2$ then the sum of $\alpha(n)$ and $\beta(m)$ is unchanged.

Next let $S_{\text{old}}$ denote the admissible arcs for the “old” set of dual variables and let $S_{\text{new}}$ denote the admissible arcs for the new set of dual variables. In order to obtain the new set of admissible arcs we first delete all arcs $(n, m)$ for which $n$ is unlabelled and $m$ is labeled and after that “all” we have to do is to check if

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

for each labeled source $n$ and for each unlabeled sink $m$.

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 necessarily a maximal flow, on the new set of admissible arcs is easy to check. For suppose that the source $n$ and the sink $m$ is such that $x(n, m) > 0$. Then the source $n$ can not be unlabeled at the same time as the sink $m$ is labeled because then
condition (ii) of the labeling routine would hold and therefore the source \( n \) would also have to be labeled if the sink \( m \) is labeled. Therefore, if \( n \) is a source which is unlabeled and \( m \) is a sink such that \( x(n, m) > 0 \) then the sink \( m \) is also unlabeled and therefore by definition of the new dual variables the arc \((n, m) \in S_{\text{new}}\). If on the other hand \( n \) is a source which is labeled and \( x(n, m) > 0 \) then \((n, m) \) must belong to \( S_{\text{old}} \) and therefore the sink \( m \) also would be labeled and again we note that the arc \((n, m) \in S_{\text{new}}\). This proves that the “old” flow also is an admissible flow on the new set of admissible arcs.

11 The last part of the primal-dual solution.

After we have gone through the updating in the dual solution change routine, we go back to the labeling routine and start the labeling where we left it. Since the new set \( S_{\text{new}} \) of admissible arcs contains at least one new arc \((i, j)\) say, such that the source \( i \) is labeled, but sink \( j \) is not yet labeled we can continue our labeling. If the new sink which we label is deficient then we will immediately obtain a new good path and we will go to the flow change routine. If the new sink that we label is full then we will anyhow label at least one more sink before we either go back to the dual solution change routine or the flow change routine at the termination of the labeling routine. This shows that we can only go back and forth between the labeling routine and the dual solution change routine at most \( M - 1 \) times before going to the flow-change routine.

Now suppose that the vectors \( \{a(n) > 0, \ 1 \leq n \leq N\} \) and \( \{b(m) > 0, \ 1 \leq m \leq M\} \) are integer valued. Then the flow change routine will always increase the total value of the flow by at least 1. Since the number of operations is at most of order \( O(NM^2) \) between each “visit” to the flow change routine and the total flow value is certainly bounded by

\[
\max\{c(n, m) \geq 0, \ 1 \leq n \leq N, \ 1 \leq m \leq M, \} \times \sum_{n=1}^{N} a(n)
\]

the primal-dual algorithm will certainly terminate in less than \( C \times NM^2 \) operations, where \( C \) is independent of \( N \) and \( M \).

If all parameters of the problem are rational it is also clear that the algorithm must end in a finite time, since we can simply multiply everything by the largest common divisor to obtain a problem with integer parameters. Finally if some or more of the parameters are real-valued it is not obvious how one proves that the algorithm must terminate in a finite time. However since one clearly can find arbitrarily close approximate solutions to the solution of
the primal problem by approximating the parameters by rational numbers in such a way that all costs are slightly less, one can use the primal-dual algorithm to find feasible dual solutions arbitrarily close to the primal solutions. Thus except for a formal proof of the proposition stated above (Proposition 8.1) we have in fact proved the equality between the solution of the primal and the dual formulation of the transportation problem.

12 Some further remarks on the primal-dual algorithm.

In this section we shall make some further remarks concerning the primal-dual algorithm. As we already have indicated we are going to use the primal-dual algorithm when computing the Kantorovich distance between images. This will imply that we will apply the primal-dual algorithm to very large transportation problems. As we shall see the number of sources, \( N \), and sinks, \( M \), for two 256 \( \times \) 256 images will be roughly \( \frac{65536}{2} \approx 32750 \) in case we want to compute the Kantorovich distance when the underlying distance function is a metric. Let us therefore go through the different steps of the primal-dual algorithm having a transportation problem of this size in mind.

First if we consider the computations involved to compute the initial dual variables \( \alpha(n), 1 \leq n \leq N \) (see section 6), we note that for each variable we have to find the least of \( M \) elements, \( M = \text{the number of sinks} \) which takes \( O(M) \) time. Thus a rough estimate of the time to determine the initial dual variables \( \alpha(n), 1 \leq n \leq N \) is \( O(NM) \). However if the cost-function \( c(n, m) \) has some structure, as it will in our applications, one can reduce the time to order \( O(N) \).

Next considering the computations involved to compute the dual variables \( \beta(m), 1 \leq m \leq M \), as we defined them in section 6, in case the cost function has no structure the time to compute all these variables would in general be of order \( O(NM) \). Even if the cost-function has some nice metric properties computing the vector \( \beta(m), 1 \leq m \leq M \) will take longer time than computing the dual variables \( \alpha(n), 1 \leq n \leq N \) since before we can compute the dual variables \( \beta(m), 1 \leq m \leq M \) we have to make a few minor changes in the elements of the cost matrix \( \{c(n, m), 1 \leq n \leq N, 1 \leq m \leq M\} \) thereby losing some of the nice structure of the cost-function. However the computation time will still be of order \( O(M) \) (with a somewhat “bigger” \( O \)).

In principal it is essentially impossible to store a "cost-matrix" of size \( 32000 \times 32000 \) since that would require approximately 8 times one giga-byte, where 8 is the number of bites required to represent a grey value. Therefore
it is necessary to have the cost-matrix given by a formula. This we will have when we compute the Kantorovich distance.

The next step in the procedure is to define an initial flow. Of course, if we define the initial flow identically equal to zero, then the time to find an initial flow is of order $O(N)$. However the price we have to pay for such a choice of initial flow is that we may have to go through the labeling routine many times extra, and most certainly will have to use the flow change routine many times extra.

To find the initial flow as we have defined it in section 6, takes essentially the same amount of time as finding the initial dual variables, that is $O(NM)$ if the cost-matrix has no structure, and $O(M)$, if the cost-matrix has some structure.

Next let us consider the flow-change routine. This routine is usually quite fast (of order $O(L)$) and usually the path length $L$ is a small number.

Before we discuss the most difficult subroutine to analyse namely the labeling routine, let us consider the dual solution change routine. In order to compute the value $\delta$ we have to make $O(N_1M_2)$ operations where $N_1$ denotes the number of labeled sources and $M_2$ denotes the number of unlabeled sinks. Usually the size of the number of labeled sources and labeled sinks are approximately the same which means that the number of operations required to compute $\delta$ is $O(NM)$. However, when we have in integer-valued cost-function and both the constrain vectors $\{a(n) : 1 \leq n \leq N\}$ and $\{b(m) : 1 \leq m \leq M\}$ have integer valued elements then one can always take $\delta = 1$.

After having computed the value of $\delta$ we have to update the present dual solution which takes $O(N)+O(M)$ operations. The time consuming part of the dual solution change routine is to determine the new set of admissible arcs. In principal, to compute the new set of admissible arcs we have to make $O(N_1M_2) + O(N_2M_1)$ operations, where as before $N_1$ denotes the number of labeled sources, $N_2$ denotes the number of unlabeled sources, $M_1$ denotes the number of labeled sinks and $M_2$ denotes the number of unlabeled sinks.

However, as we shall see later in this paper, if we compute the Kantorovich metric using the Manhattan metric as underlying metric for the distance between pixels, then the number of operations needed to find the new set of admissible arcs will be of the order $O(N_1)$ but with a fairly "large $O$".

A similar result can, in essentially the same way, be proved when we use the box-metric (the $L^\infty$ metric), but also for approximate Euclidean distance-functions it seems possible to prove that one can reduce the number of operations needed to find the new admissible arcs. Moreover, also if one as distance-function uses the square of the Euclidian distance, we have found a condition which reduces the number of operations needed to compute the new number of admissible arcs considerably. Whether this condition
always is satisfied is an open question.

Next, let us discuss the labeling routine briefly. First of all there is some ambiguity in the labeling process, since the order one chooses for which source or sink one wants to check next for condition (i) or (ii) influences the labeling. What one would like to achieve is a labeling which as a result gives a good path which accomplishes as much increase in the total flow as possible each time, (the best path so to speak). In order to improve the result of the labeling routine one therefore could modify the routine in such a way that one tries to update the labeling in such a way that when a breakthrough occurs the value of the parameter $\theta$ which occurs in the flow change routine is as large as possible.

In the algorithm for computing the Kantorovich distance, we have managed to decrease the total computation time by a factor of approximately 3, simply by continuing the labeling as long as is conceivable, that is we do not stop immediately we obtain a breakthrough but continue until we have obtained as many good paths that our choice of labeling order has made possible. Instead of just applying the flow change routine once between each time we use the labeling routine, we use the flow change routine for each good path from a deficient source to a deficient sink that the labeling routine has given us. It may happen that the parameter $\theta$ in the flow change routine will take the value 0, because an earlier use of the flow-change routine has changed the flow values of a common part of two paths, but this is easy to check.

Estimating the number of operations each time one uses the labeling routine from a totally unlabeled situation is not so easy. It depends very much on the number of admissible arcs. If there are many admissible arcs than the labeling routine can take a long time. A rough estimate is that the number of operations is of the order of the number of admissible arcs. One drawback with using the Manhattan metric as the underlying distance-function between pixels when computing the Kantorovich distance, is that the number of admissible arcs will be quite large. In the example presented later (see section 20) in which we consider two 256 x 256 images which are fairly close to each other the number of admissible arcs for the optimal dual solution will be as large as 3,000,000.

Finally, what one also would like to estimate is the number of times one uses the labeling routine from an unlabeled starting situation, since going through the labeling routine is quite time consuming. This number varies quite a lot between different underlying distance-function, and reflects in fact the different grade of sharpness for the Kantorovich distance which one obtains for different choices of underlying distance-function. The courser the underlying distance-function is, the stronger will the conditions be on
neighbouring dual variables, and the fewer iterations will be needed to reach an optimal flow.

13 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 [16] closely.

Let \((U, \delta)\) be a separable metric space, let \(P_1\) and \(P_2\), be two Borel probability measures on \((U, \delta)\) 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

\[
\text{Lip}(U) = \{ f : U \to \mathbb{R} : |f(x) - f(y)| \leq \delta(x, y) \}
\]

and define

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

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 due to the fact that, as we already have pointed out in the introduction, Hutchinson in his paper [7] uses this metric when proving a special case of an old theorem in probability theory proved for the first time by Doeblin and Fortet [6] almost 60 years ago in a more general setting).

The following duality theorem goes back to Kantorovich.

**Theorem 13.1 (Kantorovich [10].)** If \(U\) is compact then

\[
A_\delta(P_1, P_2) = B_\delta(P_1, P_2).
\]

Duality theorems similar to Theorem 13.1 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 [16] and references therein.

As mentioned in the introduction a much more general duality theorem than Theorem 13.1 is proved in [16] (Theorem 3).

14 Computing the Kantorovich distance for images. The formulation.

In the sections 5 to 11 we have in detail described the primal-dual algorithm for computing the balanced transportation problem. In the forthcoming sections we shall describe the modifications and the simplifications we have made in order to use this algorithm for computing the Kantorovich distance for images.

Let us first recall the formulation of the Kantorovich metric as the solution of a transportation problem. 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. Also assume that the two images have equal total grey value. Next let \( \Lambda \) denote the set of all non-negative mappings \( m(i, j, x, y) \) from \( K_1 \times K_2 \rightarrow R^+ \) such that

\[
\sum_{(x, y) \in K_2} m(i, j, x, y) = p(i, j) : (i, j) \in K_1
\]

and

\[
\sum_{(i, j) \in K_1} m(i, j, x, y) = q(x, y) : (x, y) \in K_2.
\]

Let \( d(i, j, x, y) \) denote a distance-function between pixels in the set \( K_1 \) and the set \( K_2 \). The Kantorovich distance \( d_K(P, Q) \) between the images \( P \) and \( Q \) is then equal to

\[
\min \{ \sum_{i,j,x,y} m(i, j, x, y) \times d(i, j, x, y) : m(\cdot, \cdot, \cdot) \in \Lambda(P, Q) \}.
\]

The dual formulation of this minimization problem was given in section 4 as follows:
Maximize \( \sum_{(i,j) \in K_1} \alpha(i,j) \times p(i,j) + \sum_{(x,y) \in K_2} \beta(x,y) \times q(x,y) \)

\( \text{when} \)

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

Our purpose with the following five sections is to use the primal-dual algorithm for computing the Kantorovich distance.

15 Computing the Kantorovich distance for images. Normalization.

As we pointed out in the definition of the Kantorovich distance, the Kantorovich distance is defined for images with equal total grey value. Therefore if we have images with unequal total grey value and want to use the Kantorovich distance to compare images, it is necessary to normalize the images so that they get the same total grey value. This can for example be accomplished by dividing each pixel value by the sum of all pixel values, thereby obtaining images of total grey value equal 1.

However since we have restricted our present computer program to integer valued parameters we can not use this normalization. Instead we simply do as follows. We first compute the total value of each picture, say \( L(P) \) for the image \( P \), and \( L(Q) \) for the image \( Q \). We then compute the largest common divisor \( G(P,Q) \) say, of \( L(P) \) and \( L(Q) \), define \( L_1(P) = L(P)/G(P,Q) \) and \( L_1(Q) = L(Q)/G(P,Q) \), and define \( \overline{P} = \{ \overline{p}(i,j) : (i,j) \in K_1 \} \) and \( \overline{Q} = \{ \overline{q}(x,y) : (x,y) \in K_2 \} \) by

\( \overline{p}(i,j) = L_1(Q) \times p(i,j) : (i,j) \in K_1 \)

and

\( \overline{q}(x,y) = L_1(P) \times q(x,y) : (x,y) \in K_2 \)

respectively.

If \( L(P) \) and \( L(Q) \) are relative prime then the total grey value of \( \overline{P} \) and \( \overline{Q} \) can be very large; therefore a few minor modifications of the two original images is sometimes appropriate to do, before one performs the normalization.

From now on we assume that the images \( P \) and \( Q \) have equal total grey value unless we say otherwise.
16 The subtraction step.

If our choice of underlying distance-function is a metric, then we can start our computation by subtracting the images "from each other". More formally we proceed as follows.

Let \( P = \{ p(i, j) : (i, j) \in K_1 \} \) and \( Q = \{ q(i, j) : (i, j) \in K_1 \} \) be two given images defined on the same set of pixels \( K_1 \), and having the same total grey value. Define

\[
P^* = \{ P^*(i, j) = \max(0, p(i, j) - q(i, j)) : (i, j) \in K_1 \},
\]

\[
Q^* = \{ Q^*(i, j) = \max(0, q(i, j) - p(i, j)) : (i, j) \in K_1 \}.
\]

and

\[
R = \{ R(i, j) = \min(p(i, j), q(i, j)) : (i, j) \in K_1 \}.
\]

It is trivial to verify that

\[
R = P - P^* = Q - Q^*
\]

from which also follows that

\[
P - Q = P^* - Q^*
\]

which because of Theorem 13.1 implies that

\[
d_K(P, Q) = d_K(P^*, Q^*)
\]

in case the underlying distance-function \( d(i, j, x, y) \) is a metric. Therefore, when computing the Kantorovich distance between two images \( P \) and \( Q \) of the same total grey value, in case we use a metric as underlying distance-function, a natural first step is to perform the "subtractions" (9) and (10) above, thereby obtaining the images \( P^* \) and \( Q^* \), and then use the primal-dual algorithm to compute the Kantorovich distance between \( P^* \) and \( Q^* \).

17 Finding admissable arcs when the underlying distance-function is the Manhattan metric.

As we pointed out when we defined the Kantorovich distance, 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-functions is the so called Manhattan-metric

\[ d(i, j, x, y) = |i - x| + |j - y| \]

or the \( l_1 \) metric, as it is also called.

One obvious disadvantage with the Manhattan distance-function is that it is not rotational invariant. Another disadvantage is that it turns out, when computing the Kantorovich distance, that the number of admissible arcs will be large compared to what an approximate Euclidian metric gives rise to, and very large compared to what an underlying distance-function defined as the square of the Euclidian metric gives rise to. A third disadvantage is that the Manhattan distance-function gives rise to a Kantorovich distance which seems to be rather course as a discriminator between images.

One advantage with the Manhattan distance-function is that it is a metric and therefore as we have pointed out above the Kantorovich distance becomes a metric.

Another advantage with the Manhatten metric is that there is an easy algorithm by which one can find new admissible arcs, and the main purpose of this section is to show how this algorithm is obtained.

Let us first prove the following proposition.

**Proposition 17.1** 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

\[ d(i, j, x, y) - \alpha(i, j) - \beta(x, y) = 0, \quad (12) \]

and similarly that for each pixel \( (x, y) \in K_2 \), there exists a pixel \( (i, j) \in K_1 \), such that (12) holds. Then, if \( (i_1, j_1) \in K_1 \) and \( (i_2, j_2) \in K_1 \),

\[ |\alpha(i_1, j_1) - \alpha(i_2, j_2)| \leq d(i_1, j_1, i_2, j_2), \quad (13) \]

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

\[ |\beta(x_1, y_1) - \beta(x_2, y_2)| \leq d(x_1, y_1, x_2, y_2). \quad (14) \]

**Proof.** Let us prove (14). 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

\[ \beta(x_1, y_1) - \beta(x_2, y_2) = \]

\[ \beta(x_1, y_1) - d(i_2, j_2, x_2, y_2) + \alpha(i_2, j_2) \leq \]

30
\[ d(i_2, j_2, x_1, y_1) - \alpha(i_2, j_2) - d(i_2, j_2, x_2, y_2) + \alpha(i_2, j_2) \leq d(x_1, y_1, x_2, y_2) \]

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

Before we state and prove the next lemma let us introduce some convenient terminology.

Let \((i, j)\) be a pixel in the support \(K_1\) of the image \(P\) and let \(\alpha(i, j)\) be a newly updated dual variable corresponding to the pixel \((i, j)\). Let \((x, y)\) be a pixel in the support of the image \(Q\). 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. If both \((x_1, y_1)\) and \((x_2, y_2)\) are low with respect to the pixel \((i, j)\) but

\[ d(i, j, x_2, y_2) - \alpha(i, j) - \beta(x_2, y_2) > d(i, j, x_1, y_1) - \alpha(i, j) - \beta(x_1, y_1) \]

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. Thus let \((x_1, y_1)\) and \((x_2, y_2)\) be two pixels. If a) \(x_1 \leq x_2\) and \(y_1 \leq y_2\) then we say that \((x_2, y_2)\) is northeast (NE) of \((x_1, y_1)\). If b) \(x_1 \geq x_2\) and \(y_1 \leq y_2\) then we say that \((x_2, y_2)\) is northwest (NW) of \((x_1, y_1)\). If c) \(x_1 \leq x_2\) and \(y_1 \geq y_2\) then we say that \((x_2, y_2)\) is southeast (SE) of \((x_1, y_1)\). Finally if d) \(x_1 \geq x_2\) and \(y_1 \geq y_2\) then we say that \((x_2, y_2)\) is southwest (SW) of \((x_1, y_1)\).

The purpose of the next lemma is to show that in case we use the \(l_1\)-metric the search for new admissable arcs can be made quite efficient.

\textbf{Lemma 17.1} Suppose that the distance-function we are using is the \(l_1\)-metric (the Manhattan-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 \((x_1, y_1) \in K_2\) and that \((x_1, y_1)\) is low with respect to \((i, j)\).

Then
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 variabel \(\beta(x, y)\) is such that 

\[-d(i, j, x, y) + \alpha(i, j) + f3(x, y) = 0.\]

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. QED.

A geometric way to look at this lemma is the following. We know from Proposition 16.1 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 admissable 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 is 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 \((i, j)\).

18 Finding admissable arcs when the underlying distance-function is not the Manhattan metric.

In case the underlying distance-function is the box-metric defined by

\[d(i, j, x, y) = \max\{|i - x|, |j - y|\}\]
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 Manhattan-metric and the box-metric (which is a good way to find approximations of the Euclidean metric) it ought to be possible to prove lemmas similar to Lemma 17.1.

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

$$d(i,j,x,y) = (i-x)^2 + (j-y)^2$$  (15)

then it does not seem so easy to prove a lemma analogous to Lemma 17.1.

To define the distance-function by (15) 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 Manhattan metric, or an approximate Euclidean metric as underlying distance-function. Thirdly this choice of distance-function seems to lead to a Kantorovich distance between images which is somewhat sharper than for example what the Manhattan or box-metric gives rise to. Moreover, if, after we have computed the Kantorovich distance, we take the square root, then we obtain a metric, a fact which we pointed out in section 13.

Since the choice of (15) for defining the underlying distance-function is quite attractive, it would of course be desirable if a lemma similar to Lemma 17.1 could be proved also 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 (15). 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 \((i_3, j_3)\) on the same line as \((i_1, j_1)\) and \((i_2, j_2)\) and 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 condition.

**Condition 18.1** Let \((i,j)\) be a pixel in \(K_1\), and let \(\alpha(i,j)\) be an admissible dual variable at the pixel \((i,j)\) 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 \((x_1, y_1)\) 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 \((x_1, y_1)\) then all pixels \((x, y_1)\) \(\in K_2\) and which are \(W\) of \((x_2, y_1)\) will also be low.

We have not been able to show that Condition 18.1 always holds, but so far, it has been satisfied in all our computer experiments.

Now, just as Lemma 17.1 makes it possible to construct an algorithm by which we can speed up the search for new admissible arcs when one uses the Manhattan metric, by assuming that Condition 18.1 holds one can introduce a stopping criteria for each line when one is looking for new admissible arcs.

19 Obtaining arc-minimal solutions.

One drawback with the primal-dual algorithm when applied to the computation of the Kantorovich distance for images is that the optimal solution obtained by the algorithm will in general not be "arc-minimal". By an arc-

minimal solution we mean a solution such that the corresponding transportation image

\[
T(P, Q) = \{((i_n, j_n), (x_n, y_n), m_n) : 1 \leq n \leq N\}
\]

from \(P\) to \(Q\) is such that if

\[
T'(P, Q) = \{((i'_n, j'_n), (x'_n, y'_n), m'_n) : 1 \leq n \leq N'\}
\]

is some other transportation image from \(P\) to \(Q\) then \(N \leq N'\).

In particular when we use the Manhattan-metric as underlying distance-function the optimal solution obtained by the algorithm described above may be such that the number of arcs is maybe 15 % to 25 % larger than the number of arcs needed in an optimal solution.

Since we are not only interested in computing the Kantorovich distance between images, but also interested in analysing the transportation images between two images, in particular as a tool for compression, we often want an optimal solution with as few arcs as possible.

However, given any optimal solution

\[
\{((i_n, j_n), (x_n, y_n), m_n) : 1 \leq n \leq N\}
\]
it is not difficult to construct an algorithm by which one can transform the present optimal solution to another optimal solution without cycles, where we by a cycle mean a path

\[ \{(i_l, j_l), (x_l, y_l), 1 \leq l \leq L \} \]

such that
1) \( L \) is even and \( \geq 4 \),
and
2) \((i_1, j_1) = (i_L, j_L)\).

Even if an optimal solution has no cycles, it is not certain that there cannot be other solutions with fewer rays. However it is not hard to prove that an optimal solution with no cycles has at most \( N+M-1 \) positive arcs where as before \( N \) and \( M \) denote the number of pixels of the images \( P \) and \( Q \) respectively.

20 Example

We shall in this paper be content with describing and illustrating the Kantorovich distance with one example, namely an example in which we have computed the Kantorovich distance between the well known image of Lenna of size \( 256 \times 256 \) (see Figure 1 in the appendix), which we denote by \( P \), and an approximation of the Lenna-image obtained from a block-based fractal coder, which we denote by \( Q \) (see Figure 2 in the appendix). (We have chosen to gather all figures in an appendix at the end of the paper, in order to keep the text on the one hand and the figures on the other hand together.)

As underlying distance-function in our example we have used the Manhattan metric.

We first changed the grey value in approximately one hundred pixels by one unit, so that the two images obtained the same total grey value. Thereafter we have subtracted 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 \( Q^* \) (see Figures 3 and 4 in the appendix). From Figures 3 and 4 we note that the coded picture did not manage to approximate neither the edges of the mirror nor the feather of the hat of Lenna very well.

As we pointed out in section 15, these images can also be defined by

\[ P^*(i, j) = \max(P(i, j) - Q(i, j), 0) \]

35
and

\[ Q^*(i, j) = \max(Q(i, j) - P(i, j), 0). \]

And as we also pointed out in section 15, in case the underlying distance-function is a metric, then the Kantorovich distance between \( P \) and \( Q \) is equal to the Kantorovich distance between \( P^* \) and \( Q^* \).

---

Figure 5. The arcs of positive length for an optimal transportation image from an original "Lenna-image" of size 256 x 256 to an approximative, fractal coded, image of Lenna.

In the figure above (Figure 5) we show the positive arcs of an optimal transportation plan between \( P^* \) and \( Q^* \) consisting of 57014 arcs, obtained after
we have used the “uncycling”-routine. (We show the same figure, Figure 5, also in the appendix.)

In this example the total grey value of $P^*$ and $Q^*$ is equal to 232161, the number of non-zero pixel values in $P^*$ and $Q^*$ are 29990 and 30679 respectively, which gives a total of 60669 pixels. The fact that the optimal solution “only” has 57014 arcs depends on the fact that several arcs are between pixels for which the grey value of the pixel in $P^*$ is the same as the grey value of the corresponding pixel in $Q^*$.

The computation time to compute the Kantorovich distance on a Sun4/69 was for this example about one hour, and the computed value was equal to 1526233. The number of arcs in the optimal transportation plan obtained by the computer programme was as large as 71453 which is at least 10785 more than needed.

The number of admissible arcs for the optimal solution was for this example quite large namely approximately 3,000,000. This reflects the fact that the set of optimal solutions to the transportation problem is probably very large, and also reflects the fact that the Manhattan metric does not give rise to a Kantorovich distance which is as discriminating between images as one would wish.

Just as is the case with the images depicted in Figures 3 and 4, (see the appendix) also this image can be regarded as a kind of difference image, and it should also from this image 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.

21 A few comments on the choice of the underlying distance-function.

Before one can compute the Kantorovich distance between images one has to choose an underlying distance-function. In this section we shall very briefly discuss four choices of distance-functions namely the box-metric

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

the Manhattan metric

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

an approximation of the Euclidean distance

$$d(i, j, x, y) = 3 \cdot \max\{|i - x|, |j - y|\} + 2 \cdot (|i - x| + |j - y|).$$
and the square of the Euclidean distance
\[ d(i, j, x, y) = (i - x)^2 + (j - y)^2. \]

The box-metric is the metric which has least discrimination ability and therefore leads to a Kantorovich distance with comparatively poor discrimination ability. Moreover the number of admissible arcs will be very large. Furthermore the number of optimal flows will also be quite large.

One good thing with this metric is that it generates a Kantorovich distance which is faster to compute than any other choice of distance-function. Next regarding the Manhattan distance-function\( d(i, j, x, y) = |i - x| + |j - y| \), it leads to a Kantorovich distance which is somewhat sharper than what one obtains if one uses the box-metric. The number of admissible is still quite large but not quite as large as if one uses the box-metric. The number of optimal transportation plans will again be quite large. The computation time is about 50 \% longer than for the box-metric.

Next regarding the approximate Euclidean distance-function
\[ d(i, j, x, y) = 3 \cdot \max\{|i - x|, |j - y|\} + 2 \cdot (|i - x| + |j - y|) \]
the main advantage is that this distance-function leads to a Kantorovich distance which is substantially sharper, so to speak, than what one obtains with the box-metric or the Manhattan-metric. The number of admissible arcs is about 1/2 of the number one obtains when one uses the Manhattan metric.

One minor drawback is that the basic unit between pixels is 5 and therefore the integers in the cost-matrix for the transportation problem becomes 5 times larger which implies that the computation time goes up. But that the computation time becomes longer is the price one has to pay for a more precise underlying distance-function. The computation time is approximately 3 times longer than for the Manhattan metric.

Another drawback is that it is more complicated to formulate and prove a lemma corresponding to Lemma 17.1. This is somewhat easier to do if one as an approximate Euclidean distance uses the distance-function
\[ d(i, j, x, y) = \max\{|i - x|, |j - y|\} + |i - x| + |j - y|. \]

Finally if we consider the distance-function
\[ d(i, j, x, y) = (i - x)^2 + (j - y)^2 \]
the main advantage is that the number of admissible arcs decreases substantially, and our experiments indicate that the number of admissible arcs is...
about 1/10:th of what one obtains if one uses the Manhattan metric. The number of optimal solutions will also be much smaller.

Another advantage is that it is rotationally invariant. Moreover as we have pointed out above, by taking the square root of the Kantorovich distance we also in this case obtain a metric between images.

One drawback is that one can not begin the computation of the Kantorovich distance by "subtracting" the images from each other as described in section 16, since the distance-function is not a metric. Another drawback is that it is harder to have control over the dual variables, and it is by no means certain that this distance-function does imply that Condition 18.1 will hold.

But as we have pointed out above one can always fairly easy check whether a flow obtained by the primal-dual solution is optimal, since it is easy (but time-consuming) to check that the set of dual variables one has obtained is in fact a set of admissible dual variables.

The computation time when using the distance-function \( d(i,j,x,y) = (i - x)^2 + (j - y)^2 \) is approximately the same as when using the distance-function \( d(i,j,x,y) = 3 \cdot \max\{|i - x|, |j - y|\} + 2 \cdot (|i - x| + |j - y|) \) that is approximately 3 times longer than when using the Manhattan metric.

22 Using transportation images for coding.

In the introduction we indicated that it could be possible to use a transportation image for coding - or rather, a quantized version of a transportation image.

The idea is based on the following observation. Let

\[ T = \{(i_n,j_n),(x_n,y_n),m_n): 1 \leq n \leq N\} \]

be an optimal transportation image, that is a transportation image which minimizes the transportation cost between two images \( P \) and \( Q \). Assume also that the transportation image \( T \) has no cycles and that the arcs are ordered in such a way that if \( n > k \) then

\[ d(i_n,j_n,x_n,y_n)*m_n \leq d(i_k,j_k,x_k,y_k)*m_k. \]

Next let us define \( N(p) \) for \( 0 < p < 1 \) by

\[ N(p) = \min\{M: \sum_{n=1}^{M} d(i_n,j_n,x_n,y_n) \times m_n > p \times d_K(P,Q)\}. \]

Now if we consider the example presented in section 20 and choose \( P \) and \( Q \) equal to \( P^* \) and \( Q^* \) respectively as they are defined in section 20, it turns out that \( N(.5) = 4337, N(.75) = 12598, N(.9) = 33875, N(.95) = 45081, \) and \( N(.99) = 54696. \)
From these numbers we thus see that for this example, half of the distance between \( P \) and \( Q \) is so to speak "carried" by 8% of the arcs.

Consequently, 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 we can construct a new approximating image \( Q(p) \) say, whose Kantorovich distance to the original Lenna is \((1 - p) \times d_K(P, Q)\). More formally let

\[
T(p) = \{(i_n, j_n, (x_n, y_n), m_n) : 1 \leq n \leq N(p)\},
\]

let \( P(p) \) be the transmitting image of the transportation image \( T(p) \), let \( Q(p) \) be the receiving image of the transportation image \( T(p) \) and define

\[
Q''(p) = Q - Q(p) + P(p).
\]

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

Similarly we can define an image

\[
P''(p) = P - P(p) + Q(p)
\]

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

In Figure 6 in the appendix, we have depicted the arcs of the transportation image \( T(1/2) \). This part of the original transportation image thus contains most of the longer arcs of the transportation image. Moreover we notice that there is quite a lot of structure in this image, and therefore it seems possible that one should be able to compress this part of the transportation image substantially for example by using vector quantization.

In Figures 7 and 8 we have depicted the images of \( Q''(p) \) and \( P''(p) \) when \( p = 1/2 \). Both these two images have the same Kantorovich distance to both the original Lenna and the fractal coded approximation. It seems clear however that the image \( P''(p) \) looks closer to the original Lenna than what \( Q''(p) \) does. The reason for this is that much of the change due to the truncated transportation image is allocated to the edge of the mirror and the edge of the hat in the mirror, and not very much is located to the face of Lenna or to the feather of the hat of Lenna, where much of the discrepancies between the original Lenna and the fractal coded Lenna is, as observed by a human eye. The eyes in the different images of Lenna are two objects where a human eye can find much discrepancy between the images.

In Figure 9 we have depicted the image \( Q''(p) \) when \( p = 3/4 \), and this time there is a considerable improvement of the quality, but it is doubtful
whether this image looks closer to the original image than the image $P''(p)$ with $p = 1/2$ which we depicted in Figure 7.

These images suggest that a simple truncation together with a coding of the remaining transportation image will not be useful as an improved image coder. A straightforward vector quantization of the transportation image might perhaps be more efficient.

23 Using transportation images for interpolation.

There are several ways in which one can use transportation images for interpolation. One way is simply to define $Q''(p)$ as we did in the previous section.

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

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

be a transportation image between the “subtracted” images $P^*$ and $Q^*$ and define the transportation image

$$ T^r = \{((i_n^r, j_n^r), (x_n^r, y_n^r), m_n^r) : 1 \leq n \leq N\} $$

by $i_n^r = i_n$, $j_n^r = j_n$, $x_n^r = i_n + \lfloor r \times (x_n - i_n) \rfloor$, $y_n^r = j_n + \lfloor r \times (y_n - j_n) \rfloor$, and $m_n^r = \lfloor r \times m_n \rfloor$, where the operation $\lfloor \cdot \rfloor$ 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 $Q$.

24 The mean flow vector field of the transportation image.

In this section we shall introduce the notion of the mean flow vector field of the transportation image and the deviation of the flow vector field of a transportation image.

The mean flow at a transmitting pixel $(i, j)$ of a transportation image

$$ T = \{((i_n, j_n), (x_n, y_n), m_n) : 1 \leq n \leq N\} $$
is defined simply as that vector \((u, v)\) which is defined by

\[
    u = \left( \frac{\sum (x_n - i \times m_n)}{\sum m_n} \right)
\]

\[
    v = \left( \frac{\sum (y_n - j \times m_n)}{\sum m_n} \right)
\]

where the sums are taken over all arcs in the transportation image which have \((i, j)\) as transmitting pixel.

Similarly we can define the mean flow at a receiving pixel.

The mean flow vector field from the transmitting image to the receiving image is simply the union of all mean flow vectors from pixels in the transmitting image, and similarly we define the mean flow vector field from the receiving image as the union of the mean flows from pixels in the receiving image.

The deviation, \(r\) say, of the mean flow vector \((u, v)\) at a transmitting pixel \((i, j)\) we define by

\[
    r^2 = \left( \frac{\sum ((x_n - u)^2 + (y_n - v)^2 \times m_n)}{\sum m_n} \right)
\]

where again the sums are taken over all arcs in the transportation image which have \((i, j)\) as transmitting pixel.

The deviation at a receiving pixel \((x, y)\) is defined similarly.

The mean flow vector field of a transportation image can be used to obtain what one could call a first order approximation of a transportation image, and the mean flow vector field of a transportation image together with the deviation of the flow, can be used to obtain what could be regarded as a second order approximation of a transportation image.

25 Using the transportation image for identifying location of 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 difference 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. If these two images are disjoint we
The algorithm of Atkinson and Vaidya presented in [2], is also based on the primal-dual algorithm. They use so called bit-scaling (see e.g [1], section 3.3) to reduce the size of the masses, and by doubling the optimal flow on the previous scale, they obtain good initial flows on the next bit-scale. They essentially only work with good paths which are positive, so they need not store all admissible arcs. When they have found a maximum flow on the present set of admissible arcs and want to find the minimum length from a labeled P-pixel to an unlabeled Q-pixel, they use Voronoi-diagrams together with a clever updating method of the dual variables already used by Vadyia in the paper [21]. They also use the Voronoi-diagrams to find new admissible arcs, and they do not determine more new admissible arcs then they, so to speak, have to. It is when computing the Voronoi-diagrams that they use the structure of the cost-function in order to lower the computational complexity from $O(N^2)$ in the general case, to $O(N)$ in case of the Manhattan metric, and to $O(N^{1.5})$ in case of the Euclidean metric. In our algorithm we instead always change our dual variables by $+1$ or $-1$, when we have to change them, and therefore we do not have to do this computation which saves time. On the other hand we do compute all admissible cells, which takes time, although we have an efficient algorithm for doing this. This does not seem to be necessary in their method. Atkinson and Vaydia have not reported any computation times in their paper [2] so it is not yet quite clear, which method is the fastest. It is also possible that by using Lemma 17.1 and Condition 18.1 one can improve their algorithm somewhat.

In section 21 we only briefly discussed advantages and disadvantages with various choices of underlying distance-function, and we have not yet reached a conclusion which distance-function is most suitable.

We have introduced the notion of a transportation image and we have briefly indicated the possibility to use the transportation image, which one obtains as consequence of the algorithm used for computing the Kantorovich distance, as a starting object for compression and interpolation. We have introduced a few functionals of a transportation image, namely the mean flow vector field, the deviation of the mean flow vector field, both of which can be used for approximating a transportation image, and the distortion matrix is introduced as a further tool for identification of dissimilarities between two given images.

It is still to early to say whether the Kantorovich distance in some sense is better than the ordinary PSNR-measure in order to measure the distance between images. Perhaps one should not think in terms of better or worse, but rather look upon the Kantorovich distance and the transportation image generated when computing the Kantorovich distance, as a complement to the ordinary $L^2$-distance and the ordinary difference images.
One drawback with the Kantorovich metric is that one has to normalize the images first, so that they get the same total grey value, and therefore the Kantorovich does in fact not distinguish between a uniformly grey image or a uniformly black image, which in practice might turn out to be a severe limitation.

A simple way to modify the Kantorovich distance so that one obtains a distance-measure also for images not necessarily with equal total grey value is as follows.

Let \( P \) and \( Q \) have unequal total grey value and let \( P \) have less total grey value than \( Q \). Let \( \Theta(P, Q) \) be the set of all transportation images which has \( P \) as transmitting image and as receiving image has an image \( Q' \) with the property that the support of \( Q' \) is within the support \( K_2 \) say of \( Q \), and also such that for all pixels \( (x, y) \in K_2 \)

\[
Q'(x, y) \leq Q(x, y).
\]

A possible definition of a distance between \( P \) and \( Q \) could then be to add the minimum cost of transportation images in the set \( \Theta(P, Q) \) to a distance measure based on the differences of the grey values at each pixel.

Another possible modification of the Kantorovich distance is the modification introduced by Luenberger in his thesis [14] chapter section 1.3.3. What Luenburger does is to put an upper and lower bound on the dual variables and thereby he can remove the requirement that the images shall have equal total grey value. Also for this modification it seems likely that the computation algorithm described in this paper can be modified so as to compute also this distance.

27 Acknowledgements.

First of all I want to thank Dr Robert Forchheimer for his support both morally and financially (via NUTEK), and his belief in me 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 ability to listen to and comment on, my efforts. Our discussions have been invaluable. Thirdly I want to thank my colleagues in the Image Coding Group all of which always have been very helpful when I have had computer problems and in particular I want to thank Lic.Eng. Niclas Wadströmer for the image generation programmes, and whose skill in programming has been of great value to my work. I also want to thank Professor Jan-Olof Eklund for sending me a copy of Werman's thesis [20], and Professor Svante Jansson for the reference to the work of Rachev ([16]).
This paper was completed when I visited New Zealand and I want to thank the Magnusson Foundation and the National Defence Research Establishment (FOA) for supporting my visit.

Finally I want to thank the staff of the mathematical and statistical department of Canterbury University, Christchurch, for their support, friendliness and kind hospitality during my stay in New Zealand.

References


Appendix.

In this appendix we have just gathered all the figures referred to in the paper. Figure 5 is the only figure which also occurs in the text.

Figure 1. The original Lenna subsampled to 256x256.
Figure 2. An 256x256 approximation of Lenna obtained by using a fractal block coder based on triangle blocks.
Figure 3. The positive differences between the image in Figure 1 and the image in Figure 2 (in that order).
Figure 4. The positive differences between the images in Figure 2 and Figure 1 (in that order).
Figure 5. The arcs of positive lengths of an optimal transportation image from an original Lenna of size 256x256 to an approximation of Lenna.
Figure 6. The arcs of a truncation of the optimal transportation image in Figure 5, containing 50% of the distance.
Figure 7. The image is obtained by “adding” a 50% truncated optimal transportation image to the image in Figure 2.
Figure 8. The image is obtained by "subtracting" a 50% truncated optimal transportation image from the image in Figure 1.
Figure 9. The image is obtained by "adding" a 75% truncated optimal transportation image to the image in Figure 2.
Figure 10. The distortion matrix of the image in Figure 3 normalised so that the largest value is 255.
Figure 11. The distortion matrix of the image in Figure 4 normalised so that the largest value is 255.
On the computation of the Kantorovich distance for images

Abstract

The Kantorovich distance for images can be defined for grey valued images with equal total grey value. Computing the Kantorovich distance is equivalent to solving a 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, which is roughly of the order \( O(N^{2.2}) \) in case the underlying distance-function is 1) the \( L^1 \) metric, 2) an approximation of the \( L^2 \) metric or 3) the square of the \( L^2 \) metric, where \( N \) is equal to the number of pixels in the two images.

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.

Nyckelord

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