The Image Foresting Transformation

A.X. Falcão, R.A. Lotufo, G. Araujo

Relatório Técnico IC–00-12

Julho de 2000

The Image Foresting Transformation

Alexandre X. Falcão*, Roberto de A. Lotufo†, Guido Araujo‡

Abstract

In this paper, we introduce an image processing operator called *Image Foresting Transformation* (IFT). The image foresting transformation maps an input image into a graph, computes a shortest-path forest in this graph, and outputs an *annotated image*, which is basically an image and its associated forest. We describe the application of IFT to region growing, edge detection, Euclidean distance transform, geodesic distance computation, and watershed transformation. All the operators are efficiently computed using the same IFT algorithm based on the same set of parameters by changing only their meaning and values. We also present a new interactive image segmentation paradigm based on the region growing operator and discuss other applications of the IFT for boundary-based object definition and shape-based interpolation.

1 Introduction

The use of graph in computer vision and image processing has been investigated for many years now. Its motivation stems from a solid theory with many efficient algorithms. As a consequence, various graph-based approaches have been proposed for image analysis [16], image coding [15], image registration [9], data clustering [25], border detection [23], object recognition [21], image retrieval [17], distance transform computation [20], etc. In most applications, either an image is thought as a graph or a graph is defined to describe the relationship among image objects. In this paper, we propose a graph-based framework suitable for both approaches.

We claim that a few graph-based formulations can be used to design many image processing operators. For example, most of the aforementioned applications can be efficiently handled using an unique optimum graph-search operator under different formulations. To prove that, we introduce the *Image Foresting Transformation* (IFT). The image foresting transformation maps an input image into a graph, computes a shortest-path forest in this graph, and outputs an *annotated image*, which is basically an image and its associated forest. Figure 1 shows a generic scheme of an image processing operator based on the IFT.

*Institute of Computing, University of Campinas, Campinas - SP, Research supported in part by CNPq, grant #300698/98-4, and by FAPESP, grant #97/13306-6

†Faculty of Electrical Engineering and Computing, University of Campinas, Campinas - SP, Research supported in part by FAPESP, grant #97/13306-6

‡Institute of Computing, University of Campinas, Campinas - SP, Research supported in part by CNPq, grant #300156/97-9

The central idea is that most part of the problem is efficiently solved as a shortest-path forest problem, so the image processing operation itself becomes a simple task applied to the annotated image. This process works as follows.

Generically, we can think of an image as a weighted and oriented graph, where the pixels are the nodes of the graph and each ordered pair of adjacent pixels defines an arc. Different image processing operators may require different graph models. That is, different weight assignments and different adjacency relations. Alternatively, one can read image objects in place of pixels, but we will adopt a pixelwise description of the IFT in this paper. The weight assigned to each arc in the graph is a non-negative value computed based on local image properties. To compute a shortest-path forest in this graph, a set of roots (i.e. pixels) is selected together with a *path-function*, that defines a non-negative value between a root and a pixel at the end of a shortest-path from the root. From each root, simultaneously, we want to grow a shortest-path tree by assigning each pixel to the tree where its path-function value is minimum. To distinguish among trees, a label is assigned to each root and propagated to the rest of the nodes in its tree. Alternatively, rooted trees that belong to the same class can be grouped with the same label. At the end, each tree is a connected component within the image and the shortest-path forest is an optimum image partition. Finally, we create an annotated image by adding three new information for each pixel: a label that identifies its connected component within the image, its parent in the forest that leads the pixel to its correspondent root, and a path-function value that represents some global measurement for the underlying problem. At least one of these information should be relevant to complete the image processing operation.

This formulation has many advantages:

1. The IFT is a powerful tool to exploit local and global image properties and to design image processing operators;

2. It depends on the same shortest-path forest algorithm based on the same set of parameters. All we have to do is to change value and meaning of these parameters;

3. It can be computed in real time in most situations. This makes it viable to design user-assisted image processing operators;

4. One can build classes of a given operator by creating different annotated images;

5. IFT-based operators can be cascaded to build new operators;

We describe the IFT for region growing, edge detection, and distance transform in Section 2. The region growing operator allows simultaneous multiple object definition. It alone constitutes a new paradigm for interactive image segmentation. Different implementations of the edge detection operator have already been used in the past for interactive segmentation [8, 6, 7]. In fact, we are extending the main ideas reported in [8] to a general graph-based image processing operator. The distance transform operator allows fast and exact computation of the Euclidean distance transform.

Clearly, image segmentation is one of the main applications of the image foresting transformation. We could present many others IFT-based object definition operators in a single

paper, and point out that there are repeated evidences in the literature as to how object information improves image filtering [10], interpolation [18], registration [14], etc. However, the aim of this paper is to show that IFT is more than an image segmentation operator. It can be used to compute other types of image content, such as distance transforms, geodesic distances and other image/object features.

We present a shortest-path forest algorithm to compute the IFT in Section 3. Since our definition of path-function differs from the traditional sum of arc weights on the path, we present some theoretical results that prove the optimality of IFT in Section 4. We discuss in Section 5 the use of IFT for interactive image segmentation under both approaches, region-based and boundary-based, shape-based interpolation [18], watershed transformation [2], and geodesic distance computation [12]. Finally, we state our conclusions and discuss our on going research on IFT in Section 6.

2 The Image Foresting Transformation

In this section we define terms and concepts that are used in the rest of the work.

Definition 1 *An n-dimensional m-band digital image* $\mathbf{I}$ *is a pair* $(I, \vec{f})$ *consisting of a finite n-dimensional array* I *of pixels and a director function* $\vec{f}(p)$*, that assigns to each pixel* p *in* I *an m-dimensional scale-vector.*

We call $\mathbf{I}$ an $nDmB$ image, or simply an image. The scale-vector $\vec{f}$ in $\mathbf{I}$ represents any finite number of image properties. For example, in a colored image, $\vec{f}(p)$ can be defined as a 6-tuple where the entries are the values of red, green, blue, and their respective gradient magnitudes at pixel p.

Definition 2 *Let the coordinates at the center of a pixel be an n-tuple of integers in* Z^n*. We define an adjacency relation* ρ *in* Z^n *by considering all pairs of pixels* $(p, q) \in I \times I$ *satisfying* $d(p, q) \leq R$*, where* d *is the Euclidean distance between* p *and* q*, and* R *is the adjacency radius. In other words, an adjacency relation* ρ *accounts for the* ρ *closest pixels to* p *in* Z^n*.*

Figure 2 illustrates three types of adjacency relations for a 2-dimensional image. In Figures 2a and 2b, the adjacency relation accounts for the four ($R = 1$) and eight ($R = \sqrt{2}$) closest neighbors of a pixel, respectively. The adjacency relation showed in Figure 2c is less common. It takes into account the twenty closest neighbors ($R = \sqrt{5}$) of a pixel. The adjacency relation is then a way of defining local connectivity between pairs of pixels. In some applications, however, we may want to make all pixels in the image adjacent to each other.

Definition 3 *An annotated image is an image together with an associated shortest-path forest.*

Definition 4 *The Image Foresting Transformation (IFT) is a sequence of two consecutive mappings* $\mathbf{I} \rightarrow \mathbf{G} \rightarrow \mathbf{I_a}$*, where* $\mathbf{G}$ *is a graph defined in* I *and* $\mathbf{I_a}$ *is an annotated image of* $\mathbf{I}$*.*

Generically, we think of $\mathbf{I} = (I, \vec{f})$ as a weighted and oriented graph $\mathbf{G}$, where the pixels in I are the nodes of the graph and each ordered pair (p, q) of ρ-adjacent pixels in $\mathbf{G}$ defines an arc. Figure 3a shows a 2-dimensional example of $\mathbf{G}$ for $\rho = 4$.

Definition 5 *We define* $\mathbf{w}(p, q)$ *a weight function that assigns a non-negative weight to each arc* (p, q) *in* $\mathbf{G}$*, corresponding to the penalty to go from* p *to* q*.*

Given a family $\mathcal{R} = \{R_1, R_2, \ldots, R_K\}$ of K root sets (i.e. pixel sets) in $\mathbf{G}$, we assign the same label i to all roots in $R_i \in \mathcal{R}$, $i = 1, 2, \ldots, K$. From each pixel $r \in R_i$, we want to grow a tree rooted at r by propagating label i to all its nodes, such that each node in $\mathbf{G}$ is assigned to only one tree. This process is based on the concept of *path-function* $\mathbf{pf}$ defined as follows.

Definition 6 *Let* $< p_1, p_2, \ldots, p_l >$ *be the path from a root* p_1 *to a pixel* p_l *in* $\mathbf{G}$*. A function* $\mathbf{pf}(p_1, p_l)$ *is a path-function if its domain is the set of ordered nodes on the path and* $\mathbf{pf}(p_1, p_l) = \mathcal{F}(\mathbf{w}(p_1, p_2), \ldots, \mathbf{w}(p_{l-1}, p_l))$*, for some non-negative non-decreasing function* $\mathcal{F}$*.*

If $\mathbf{pf}$ is defined as:

$$\sum_{i=1}^{l-1} \mathbf{w}(p_i, p_{i+1}), \tag{1}$$

the process described above outputs a shortest-path forest as proposed by Dial [5]. However, the restriction of defining $\mathbf{pf}$ as a *non-negative non-decreasing* function is sufficient to output a shortest-path forest in $\mathbf{G}$ and we will prove that in Section 4. For the time being, we should just keep in mind that there are applications which require other types of *non-negative non-decreasing* path-functions.

The path-function $\mathbf{pf}(r, p)$ represents a penalty to go from a root r to a node p in $\mathbf{G}$. Our aim is to assign p to the tree rooted at r, where $\mathbf{pf}(r, p)$ is minimum. At the end, all nodes in the forest whose trees are rooted at the nodes in R_i are labeled i. We will have K shortest-path forests in $\mathbf{G}$, or simply a *shortest-path forest*. Figure 3b shows a 2-dimensional example of a shortest-path forest in the graph shown in Figure 3a for $\mathbf{pf}$ as in Equation 1 and $\mathcal{R} = \{\{a\}, \{f, g\}\}$. The label of each root in $\mathcal{R}$ is propagated to each node in $\mathbf{G}$. The label and the path-function value for each node are shown in Figure 3b.

Let K_r be the total number of roots $r_i \in \mathcal{R}$. Note that, each shortest-path tree $\mathbf{T}_i$, rooted at r_i, $i = 1, 2, \ldots, K_r$ is a connected component in $\mathbf{I}$ and the IFT computes an optimum partition of $\mathbf{I}$ with K_r connected components where:

$$\sum_{i=1}^{K_r} \sum_{\forall p \in \mathbf{T}_i} \mathbf{pf}(r_i, p) \tag{2}$$

is minimum. In Figure 3b, for example, this process results two connected components with label 2 and another component with label 1 forming an image partition with minimum penalty 21 according to Equation 2.

At the end, an $nD(m+3)B$ annotated image $\mathbf{I_a} = (I_a, \vec{f}_a)$ is created, where $I = I_a$ and $\vec{f}_a$ is an extension of the director function $\vec{f}$ that includes three new information for each pixel: a label that indicates its connected component within the image, the parent of the pixel in the forest that leads the pixel to its corresponding root, and a path-function value that represents some global measurement for the underlying problem (see Figure 3b). These three new image properties describe the resulting forest and then the IFT can be written as:

$$\mathbf{I_a} = IFT(\mathbf{I}, \rho, \mathbf{w}, \mathcal{R}, \mathbf{pf}). \tag{3}$$

At this point one can conclude that, to transform an image into a graph, we just need to think of pixels as nodes and use an adjacency relation to define the arcs. However, different adjacency relations will lead us to different graphs that represent the same image. Thus, what graph representation should we use for a given problem? The parameters of the IFT are always the same, but their meaning and value change for different problems. Then, how should we choose them? Finally, how can we use this formulation to solve image processing problems?

Since the answers to the questions above depend on the underlying image processing operation, we will address them in the next sections by using three examples: region growing, edge detection, and Euclidean distance transform. In each example, we will be interested in one of the three new information created by the IFT. For region growing, we are interested in the labels assigned to each pixel. For edge detection, we are interested in paths between pixels, which are obtained based on the information about the parent of each pixel in the shortest-path forest. For distance transform, we are interested in the path-function value assigned to each pixel. We expect the reader will be able to extend the IFT concepts to other examples afterwards.

2.1 Region growing

Images keep local properties between adjacent pixels that can be measured by computers with no problem. Unfortunately, the image content from the view point of the users is global. Users understand an image as a collection of regions where the similarity among pixels within the same region is high, according to some set of image properties (e.g. brightness, color, texture), and low between different regions. They also understand that groups of regions form objects in the image. Such global properties are much more difficult to be measured by computers without human help. This is probably the main motivation for interactive image segmentation.

In this section, we show how to exploit local and global similarities between pairs of pixels to find high similarity regions in the image. In section 5, we show how to use this result to build region-based object definition operators.

2.1.1 From image to graph

The goal of this section is to define arguments ρ and $\mathbf{w}$ from Equation 3, such that the graph constructed by the IFT operator can be used to solve the region growing problem.

For region growing, we think of an image $\mathbf{I} = (I, \vec{f})$ as a weighted and **non-oriented** graph $\mathbf{G}$, where the pixels in I are the nodes of the graph and each pair (p, q) of ρ-adjacent pixels defines a non-oriented arc in $\mathbf{G}$. We choose $\mathbf{w}(p, q)$ as a non-negative function of $\vec{f}(p)$ and $\vec{f}(q)$, which is proportional to the degree of dissimilarity between p and q. Three examples of $\mathbf{w}(p, q)$ are:

$$|\vec{f}(p) - \vec{f}(q)|, \tag{4}$$

$$k_1 exp(-(g(p, q) - k_2)^2/k_3), \tag{5}$$

$$\sum_{x=1}^{m} |f_x(p) - f_x(q)|/m, \tag{6}$$

where $g(p, q) = \max_{\forall x} |f_x(t) + 2f_x(p) + f_x(v) - f_x(u) - 2f_x(q) - f_x(w)|/4$ is computed based on a local neighborhood of (p, q) (see Figure 4), f_x is the x-th component of the m-dimensional scale-vector $\vec{f}$, and k_1, k_2, and k_3 are positive numbers.

Note that, one can think in various other functions, they will output different weighted graphs, and consequently, they provide different IFT results. We have not investigated yet the variety of combinations involving different image properties and weight functions for region growing. However, the weight function described in Equation 5 usually leads to better results than linear functions. It is less sensitive to noise and emphasizes better region boundaries with values of g close to k_2 (i.e. assigns higher weights to arcs (p, q) whose $g(p, q)$ is close to k_2).

The adjacency relation ρ seems to have less influence on the result of region growing operations, but it is very important to guarantee correct results in other situations. An example is the correct computation of the Euclidean distance transform, as we will see in Section 3.

2.1.2 From graph to annotated image

In the previous section we determined arguments ρ and $\mathbf{w}$ that are used by IFT to reduce region growing to an equivalent shortest-path forest problem. In this section we define the other two arguments in Equation 3, i.e. $\mathcal{R}$ and $\mathbf{pf}$, which are then used to solve the equivalent shortest-path forest problem.

Figure 5a shows a gray-scale display of a $2D3B$ image whose $\vec{f}$ is composed by the main components, red, green, and blue. Suppose we want to detect the big dark pepper on the top of this figure. Then we should have at least a root inside it (with label 1, for example) and another root outside (with label 2). The roots selected inside it are represented by white lines in Figure 5a, while black lines represent the roots selected outside. Roots selection can be manual or automatic, depending on our knowledge about the problem. In this case, we are using manual selection.

For region growing, we define the path-function $\mathbf{pf}$ as

$$\max_{\forall i \in \{1,2,...,l-1\}} \{\mathbf{w}(p_i, p_{i+1})\}, \tag{7}$$

since the result of IFT, when using this definition for $\mathbf{pf}$, matches the user expectation

much better than if Equation 1 is used. Nevertheless, there may be other functions better than this.

Figures 5b and c show the weight function at the horizontal and vertical arcs, using $\rho = 4$, and $\mathbf{w}$ as in Equation 5. Here, brightness is inversely proportional to the weight assigned to each arc. Figure 5d shows the result of extracting the big dark pepper by region growing using **pf** as in Equation 7. In this case, we are assuming the values of g (in Equation 5) along the pepper's boundary are close to $k_2 = 100$.

Alternatively, we could have set lower weights to arcs whose pixels have average color close to red, and higher weights elsewhere. Since the big dark pepper in Figure 5a is red in the original image.

In region growing, we say that a pixel is assigned to its most similar root in $\mathcal{R}$, taking into account all possible paths from all roots in $\mathcal{R}$. That is, when we decide that a pixel p belongs to the same tree rooted at r, our decision is taking into account not only the local dissimilarity between ρ-adjacent pixels, but also the dissimilarity between each pair of ρ-adjacent pixels in the shortest-path from r to p. Therefore, the path-function value is a global dissimilarity measure in the image.

2.2 Edge detection

Given a pair of pixels (p, q) on the boundary of an object in $\mathbf{I}$, we can define an *edge* as a "connected" and "oriented" curve from p to q made up of ρ-adjacent pixels on the boundary of the object.

2.2.1 From image to graph

As before, in this section we specify the IFT arguments ρ and $\mathbf{w}$, that are required to build the graph formulation for the edge detection problem. For edge detection, we think of an image $\mathbf{I} = (I, \vec{f})$ as a weighted and oriented graph $\mathbf{G}$, where the pixels in I are the nodes of the graph and each pair (p, q) of ρ-adjacent pixels defines an oriented arc, from p to q, in $\mathbf{G}$. We choose $\mathbf{w}(p, q)$ as a non-negative function of $\vec{f}(p)$ and $\vec{f}(q)$, which represents the cost of considering (p, q) as an *edge element*.

As an example, the weight function $\mathbf{w}(p, q)$ can be defined as

$$k_1(1 - exp(-(g(p, q) - k_2)^2/k_3)), \tag{8}$$

where $g(p, q) = \max_{\forall x} (f_x(t) + f_x(u) - f_x(v) - f_x(w))/2$ is computed using the neighborhood of (p, q) shown in Figure 6, f_x is the x-th component of the m-dimensional scale-vector $\vec{f}$, k_1 and k_3 are positive numbers, and k_2 is a real number. If we wish to detect edge elements such that the gradient vector $\vec{g}(p, q)$ points to the right side of (p, q), as shown in Figure 6, we should choose k_2 a negative number. That is, we assign lower weights to edge elements whose values of g are close to k_2. This will favor to detect edges with clockwise orientation around the object of interest [8].

Analogous to region growing, there are many ways of defining weight functions that lead to different IFT results. The work reported in [8] provides several examples of image properties, weight functions, and training to detect 2-dimensional closed, connected and

oriented boundaries in medical images. These techniques, together with the general framework enabled by the IFT operator, could be a start point to further investigate weight functions amenable to region growing and edge detection.

2.2.2 From graph to annotated image

To detect an edge between two specified pixels p and q on the boundary of an object, we select p as a root, the path-function **pf** as in Equation 1, and compute a forest in **G** of one tree rooted at p. Note that in this case $\mathcal{R} = \{\{p\}\}$. At the end, the path from p to q will be the shortest-path and its path-function value will represent the minimum cost value of considering this path an edge. Although both path-functions, suggested in Equations 1 and 7, result an optimum edge between p and q, Equation 1 gives the best practical solution for edge detection.

Figure 7a shows a $2D1B$ image of a wrist obtained by magnetic resonance (MR). Since it is a gray-level image, we can define f_1 as brightness in Equation 8. Figures 7b and c show the average between the weight values assigned to up- and right-oriented arcs, and to down- and left-oriented arcs, respectively, for $\rho = 4$ and $\mathbf{w}$ as in Equation 8. Here, brightness is directly proportional to weight. Figure 7d shows a clockwise oriented edge computed, using $\mathbf{pf}(r, p)$ as in Equation 1, where r is a root and p a pixel selected on the boundary of the wrist. Among all possible paths within the image, there are two promising edges between the two points r and p selected on the wrist's boundary. The IFT-based edge detection takes the longer one, because it is a clockwise oriented edge. Orientation is a powerful information that should be considered during the design of boundary-based object definition operators.

Coming back to Figure 5a, we could also use Equation 8 to detect clockwise oriented edges on the boundary of the long bright pepper on the left side of this figure. But the same is not valid for the big dark pepper on the top of Figure 5a, because the direction of $\vec{g}$ changes along its boundary. Unless, we assign lower weights to edge elements whose color on the right side is close to red, and higher weights elsewhere. In conclusion, we better define **G** as a weighted and **non-oriented** graph for edge detection, if it is impossible to model the boundary orientation information.

2.3 Distance transform

Distance transform is a powerful transformation that assigns to object pixels in a binary image their distance to the background pixels. Figure 8a shows the Euclidean distance transform on a binary image, where the values assigned to each pixel are the squared distance values. Distance transform has several applications, such as skeletons, dilation, erosion, classification, interpolation, etc. However, despite the Euclidean distance be the most natural metric for those applications, the difficulty to implement efficient algorithms has conducted researches to other metrics, such as Chanfer, chessboard, and city-block.

The IFT can be used to compute the distance transform with different metrics. We have chosen the Euclidean distance transform in this section.

2.3.1 From image to graph

For distance transform, we think of a binary image as a weighted and **non-oriented** graph $\mathbf{G}$, where the pixels are the nodes of the graph and each pair (p, q) of ρ-adjacent pixels defines a non-oriented arc in $\mathbf{G}$. As we will see in Section 3, the correctness of our Euclidean distance operator requires a special care in choosing ρ. For the time being, let's just assume ρ equal to 8 in the 2-dimensional case. The weight function $\mathbf{w}(p, q)$ is defined as:

$$|d(r, p) - d(r, q)|, \tag{9}$$

where r is a background pixel and d is the squared Euclidean distance value. Since the weight function depends on a background pixel, the weight assigned to (p, q) may be different for different background pixels. Figure 8b illustrates this situation for two background pixels r_1 and r_2. In this case, $\mathbf{w}(p, q) = 3$ based on r_1 and $\mathbf{w}(p, q) = 1$ based on r_2. Then which one should we use? The answer to this question is underway.

2.3.2 From graph to annotated image

We define the roots in $\mathcal{R}$ as the first layer of background pixels (see Figure 8a), such that K is the number of pixels in this layer and each R_i, $i = 1, 2, \ldots, K$, has one of these pixels. Note that, only one of these roots will be the background pixel taken into account to compute $\mathbf{w}(p, q)$ in Equation 9. By considering the possible displacement vectors from p to q, for $\rho = 8$, the weight function $\mathbf{w}(p, q)$ is represented as one of three vectors of positive increments $(dx(p, q), dy(p, q))$: $(0, 1)$, $(1, 0)$, or $(1, 1)$ (see Figure 9a). Thus the path-function $\mathbf{pf}$ is defined as:

$$(\sum_{i=1}^{l-1} dx(p_i, p_{i+1}))^2 + (\sum_{i=1}^{l-1} dy(p_i, p_{i+1}))^2. \tag{10}$$

Therefore, all we have to do is to accumulate the displacements dx and dy in the path from a root p_1 to a node p_l in order to evaluate at any time the path-function value at p_l using Equation 10. Figure 9b shows two paths from a root p_1 to a node p_4 with the accumulated displacements to each node. Note that, the path $< p_1, p_2, p_3, p_4 >$ where $\mathbf{pf}(p_1, p_4) = 10$ is the one with the correct Euclidean distance. This is not the unique path, but the correct Euclidean distance is always obtained by finding a path where $\mathbf{pf}$ is minimum, and this is what the *IFT* does.

3 Algorithm

IFT algorithm

Input: An $nDmB$ image $\mathbf{I}$; an adjacency relation ρ in Z^n; a non-negative weight function $\mathbf{w}$ between ρ-adjacent pixels; a family $\mathcal{R}$ of labeled root sets; a non-negative non-decreasing path-function $\mathbf{pf}$.

Output: An $nD(m+3)B$ annotated image $\mathbf{I_a}$.

Auxiliary Data Structures: An nD array lb with the current label k of each pixel; an nD array pf with the current path-function value of each pixel; an nD array d indicating for each pixel, its current parent in the forest; a priority queue Q of pixels; a list L of pixels which have already been processed.
begin

1. set $pf(p)$ to ∞, $lb(p)$ to 0, and $d(p)$ to *nil* for all pixels $p \in I$;
2. set $pf(r)$ to 0 and $lb(r)$ to its corresponding label for all labeled roots $r \in \mathcal{R}$, and put r in Q;
3. *while* Q is not empty *do*
   a. remove a pixel p from Q such that $pf(p) = \min_{p' \in Q}\{pf(p')\}$, and put p in L;
   b. *for* each pixel q, such that (p, q) are ρ-adjacent pixels and $q \notin L$ *do*
      (i) compute tmp based on $pf(p)$ and $\mathbf{w}(p, q)$, representing the path-function value to reach q passing through p;
      (ii) *if* $tmp < pf(q)$ *then*
         a. set $pf(q)$ to tmp, $lb(q)$ to $lb(p)$, $d(q)$ to p;
         b. *if* $q \notin Q$ *then* insert q in Q *else* update position of q in Q;
      endif;
   endfor;

endwhile;

end

At the end, the three new bands in the annotated image $\mathbf{I_a}$ contains the values of the nD arrays lb, pf, and d, that represent the resulting forest.

There are some important observations about the implementation of the IFT algorithm.

Clearly, the bottleneck of the IFT algorithm is in maintaining the priority queue Q. Usually, we implement Q as a binary heap which should be sufficient to guarantee the efficiency of the IFT in most applications. However, one can also take advantage of other clever solutions that exist in the literature of network flows [1]. For example, in situations we have control over the maximum weight assigned to an arc in $\mathbf{G}$, Q can be maintained in linear time if we define it as in the Dial's implementation of the Dijkstra's algorithm. We can adopt this implementation of Q for all operators described in this paper (see [7]).

Since the minimum path-function value at a pixel is not unique, it is desirable to implement the priority queue Q with a *first in first out* restriction for pixels with the same path-function value. For region growing, for example, if we have roots within a region of constant path-function value, a pixel will be assigned to the closest root in Z^n.

An observation must be made as regarding to the correctness of the Euclidean distance transform using the IFT algorithm. For Euclidean distance transform, the success of the IFT can only be guaranteed if all roots in $\mathcal{R}$ have access to all pixels in $\mathbf{I}$. Suppose, for

example, $\rho = 4$, a pixel p and its 4-adjacent pixels p_i, $i = 1, 2, 3, 4$, and three roots r_j, $j = 1, 2, 3$, as shown in Figure 10. The roots r_j, $j = 1, 2, 3$ are equidistant to p_i, $i = 1, 2$. That is, $\mathbf{pf}(r_j, p_i) = 41$, $i = 1, 2$, $j = 1, 2, 3$, by Equation 10. Then, suppose p_1 is reached by r_3 and p_2 is reached by r_2 before r_1 reaches them. According to line 3.b.(ii) of our algorithm, the path-function values in p_1 and p_2 are not updated by the time r_1 reaches them. Therefore, p will be reached by either r_2 or r_3 with path-function value equal to 52. However, r_1 is its closest root with path-function value equal to 50. This does not happen for $\rho = 8$, but it is possible to find out other cases where the exact Euclidean distance transform does not work for $\rho = 8$ [4]. In our method, we just have to set the value of $\rho = 8$, as suggested by [4], as a function of the maximum Euclidean distance in the image and the IFT will output the exact Euclidean distance transform.

Note that, the *IFT* algorithm computes Euclidean distance values for pixels inside and outside the object, simultaneously. This is an advantage in some applications as we will see in Section 5. Alternatively, the Euclidean distance computation can be restricted into the object by inserting background pixels in the list L beforehand.

4 The optimality of the *IFT*

For the edge detection operator, we have defined $\mathcal{R}$ with a single set R_1 with a single root r and **pf** as in Equation 1. In this case, the *IFT* algorithm becomes the well known Dijkstra's algorithm [3] and the output forest contains a single shortest-path tree. By defining multiple roots in $\mathcal{R}$ and the same path-function as in Equation 1, the *IFT* algorithm becomes the Dial's algorithm [5], which results a shortest-path forest. In this paper, we have extended the underlying concepts of the Dial's algorithm to claim that a shortest-path forest is guaranteed for any non-negative non-decreasing path-function **pf**. This is proved as follows.

Lemma 1 *Any non-negative non-decreasing path-function* **pf** *in the IFT algorithm can be transformed into a Dijkstra's shortest-path function as defined in Equation 1.*

Proof 1 *Let $p \neq r$ be a pixel, r a root, and path-function $\mathbf{pf}(r, p)$ the shortest-path estimate for path $r \rightsquigarrow p$. In the IFT algorithm, the only place the shortest-path estimate is updated is on line 3.b.(ii). This operation aims at relaxing constraint $\mathbf{pf}(r, p_{i+1}) \leq \mathbf{pf}(r, p_i) + \mathbf{w}'(p_i, p_{i+1})$, for some $\mathbf{w}'(p_i, p_{i+1})$, by assigning a smaller value tmp to $\mathbf{pf}(r, p_{i+1})$. Given that this constraint is an invariant for the whole execution of the algorithm, we have at any time after the start of execution $\mathbf{pf}(r, p_{i+1}) = \mathbf{pf}(r, p_i) + \mathbf{w}'(p_i, p_{i+1})$, for some node p_{i+1} adjacent to p_i. If $\mathbf{pf}(r, p_i)$ is a non-negative non-decreasing path-function, then $\mathbf{pf}(r, p_{i+1}) - \mathbf{pf}(r, p_i) \geq 0$ is an invariant and thus $\mathbf{w}'(p_i, p_{i+1}) \geq 0$ for all arcs in $\mathbf{G}$.*

Theorem 1 *If* **pf** *is defined as a non-negative non-decreasing path function, then the IFT algorithm generates a shortest-path forest where each tree has a root in $\mathcal{R}$.*

Proof 2 *First, we can create an artificial node a in $\mathbf{G}$ with one arc to each root $r \in \mathcal{R}$ and assign weight $w(a, r) = 0$. We consider a as the only root in the IFT algorithm. By Lemma 1, any non-negative non-decreasing path-function can be transformed into a*

Dijkstra's shortest-path function in the IFT algorithm. Then, the IFT algorithm run from a results the a shortest-path tree as in Dijkstra's algorithm. If we remove a from this tree, the result is a shortest-path forest where each tree has a root in $\mathcal{R}$.

Note that, by Theorem 1, the region growing and the Euclidean distance transform operators compute an optimum partition of $\mathbf{I}$ (i.e. a shortest-path forest in $\mathbf{G}$) according to Equation 2, where the path-function $\mathbf{pf}$ is given by Equations 7 and 10, respectively.

5 Results

In this section, we discuss the use of the *IFT* for interactive image segmentation, shape-based interpolation, watershed transformation, and geodesic distance computation.

5.1 Interactive image segmentation

There are many segmentation tasks that require extensive user's help. In video composition, for example, a skilled user is often required to manually extract objects from an arbitrary background of a video sequence and include them in another sequence. This is an inaccurate and imprecise task that usually takes half of the total time for video production. In medical imaging, one can find many other examples of how laborious can be user assistance in image segmentation.

Image segmentation consists of two tightly coupled tasks - *recognition* and *delineation*. Recognition is the process of identifying roughly the whereabouts of a particular object in the image and delineation is the process of specifying the precise spatial extent of this object. While computer algorithms are very effective in object delineation, the absence of relevant global object-related knowledge is the main reason for their failure in object recognition. On the other hand, a simple user assistance in object recognition is often sufficient to complement this deficiency and to complete the segmentation process.

In the next sections, we show two ways of exploiting the superior abilities of human operators (compared to computer algorithms) in object recognition and the superior abilities of computer algorithms (compared to human operators) in object delineation to develop efficient interactive image segmentation methods.

5.1.1 Boundary-based object definition

In the past, we have presented four user-steered image segmentation paradigms for boundary-based object definition: live wire, live lane, 3D live wire, and live-wire-on-the-fly [8, 6, 7]. They represent different ways of using the *IFT*-based edge detection operator, and so different implementations. Their efficiency has been proven in several medical applications [24, 11, 22, 19]. We have also used the live-wire-on-the-fly segmentation paradigm in digital video processing applications. In this section, this segmentation paradigm is described under the *IFT* framework.

In live-wire-on-the-fly [7], a 2-dimensional boundary is a closed, connected, and oriented contour made up of oriented pixel edges. In fact, the graph $\mathbf{G}$ is a little different from what

we have described here. The pixel vertices are the nodes of **G** and each oriented pixel edge defines an arc in **G**. Since this is an implementation detail, we describe the method using the standard notation of the IFT for edge detection. Therefore, we say that a 2-dimensional boundary is a closed, connected, and oriented contour made up of edges, where each edge is a shortest-path in **G** made up of oriented arcs between pairs of ρ-adjacent pixels ($\rho = 4$ or 8, where $\rho = 8$ output smoother boundaries).

To define a 2-dimensional object, the user first selects an initial point on the boundary of the object. For any subsequent point indicated by the cursor, an optimum edge from the initial point to the current point is found via IFT and displayed in real time (see Figure 11a). The user thus has a live wire on hand which is moved by moving the cursor. If the cursor goes close to the boundary, the live wire snaps onto the boundary (Figure 11b). At this point, if the live wire describes the object edge appropriately, the user deposits the cursor which now becomes the new starting point (Figure 11c) and the process continues this way until a close operation is requested by the user (Figure 11d).

Note that, object recognition is up to the user who selects points on the object's boundary and/or places the cursor close to the boundary while object delineation is computed by IFT.

5.1.2 Region-based object definition

In this section, we present a new interactive image segmentation paradigm for region-based object definition. This method represents one way of using the IFT-based region growing operator.

To define an object, the user draws roots with the same label inside the object (white line in Figure 12a) and roots with a different label in the background (black line in Figure 12a). The IFT propagates the labels splitting the image into two parts. The first part is composed by pixels which are more connected to the white roots (object) than to the black ones (background), and the other way around for the second part. The user verifies the result of this optimum image partitioning as shown in Figure 12b. By adding (or deleting) new roots inside (or outside) the object, the user can improve the previous results toward the complete object extraction (Figures 12c-f).

Again, object recognition is done by the user who draws roots inside and outside the object while object delineation is computed by IFT.

Note that, one can choose more than two labels to define multiple objects simultaneously. Moreover, the computation time of the IFT algorithm is about the same for different number of roots.

5.2 Shape-based interpolation

Three dimensional data created by tomographic medical imaging devices are usually presented as a sequence of $2D1B$ gray-level images (i.e. slices). The distance between the slices is typically greater than the distance between the pixels within the slices. An interpolation technique is often applied to convert the data into an isotropic volume with the same resolution in all three dimensions. With this aim, Raya and Udupa [18] have introduced

the shape-based interpolation which can be applied to segmented binary images. The main advantage of this approach is in situations where the user has to segment the object interactively in a slice-by-slice fashion. Interpolation before segmentation would make the user's task even more difficult.

In shape-based interpolation, each binary slice is converted into a gray-level image, in which the gray value approximates the distance of the pixel to the nearest point on the boundary of the object. Positive values are assigned to pixels within the object and negative values to pixels outside. The intermediate binary slices are estimated by interpolating the distances and thresholding the result at zero.

Distance transform is often used to convert each binary slice into a gray-level image. The traditional approach is to invert the binary slices and combine both distance transforms for object and background. The usual metric is Chanfer's distance. A better approach certainly is to compute the Euclidean distance values inside and outside the object at same time. This is what the IFT algorithm, as presented in Section 3, does. The values inside correspond to the Euclidean distance transform for the object and the values outside can be shifted to correspond the values of the Euclidean distance for the background.

Note that, shape-based interpolation is one example where two IFT operators, one for segmentation and the other for distance transform, are cascaded to build an image processing operation.

5.3 Watershed transformation

The watershed transformation [2] is a paradigm in morphological image segmentation. It is usually computed on gradient images from selected markers. When the markers are the regional minima of the gradient image, an over-segmentation problem appears due to the large number of markers. To overcome this problem, a reduced number of markers should be selected inside and outside the object. In situations where the markers are not the regional minima, the homotopy of the gradient image should be changed to impose the selected markers as the only regional minima. In [13], we showed that the watershed transformation is a particular case of the IFT, and in the IFT framework, there is no need to change the homotopy of the gradient image for arbitrary markers. In this section, we present this result by describing the watershed transformation in the IFT framework.

To compute the watershed transform, we think of an image as a weighted and oriented graph, where the pixels are the nodes of the graph and each pair of ρ-adjacent pixels defines an oriented arc. We choose $\mathbf{w}(p,q)$ such that the weight assigned to each arc (p,q) is the morphological gradient $g(q)$ computed at q. That is, all incoming arcs to a pixel q have the same weight value $g(q)$ (see Figure 13). Now, we define the path-function **pf** as in Equation 7. In this case, the IFT is equivalent to the the watershed transform with the change of homotopy [13]. If the markers (i.e. labeled roots) are selected at the regional minima of the gradient image, the arc with the greatest weight along the shortest-path (as defined by Equation 7) will always be the last arc. In this case, the position of a pixel will never be updated in the priority queue of the IFT algorithm (see Section 3), and then, it can be reduced to the watershed algorithm using an ordered queue.

5.4 Geodesic distance

The geodesic distance transform has played an important role in the design of morphological operators [12]. Examples of applications are the geodesic skeleton by influence zones and interpolation from contour lines. In this section, we describe the geodesic distance transform under the IFT framework.

Figure 14 shows two examples in a binary image where we wish to compute the geodesic distance between two pixels A and B. For that, we think of this image as an **non-oriented** graph $\mathbf{G}$, where the pixels are the nodes of the graph and each pair (p, q) of 8-adjacent pixels defines a non-oriented arc in $\mathbf{G}$. We assign $\mathbf{w}(p, q) = 1$ to all arcs, choose A as the only root in $\mathcal{R} = \{\{A\}\}$, and **pf** as in Equation 1. By inserting background pixels in the list L beforehand (see Section 3), a geodesic path from A to B is computed via the IFT algorithm (see solid line in Figure 14). There are multiple paths, but all of them with the same geodesic distance represented by the final path-function value at B.

Note that, the geodesic path is being defined here based on the chessboard metric. Its computation based on the IFT is very efficient. Figure 14 also shows the geodesic path from A to B (dashed line) based on the Euclidean distance. In this case, the appropriate IFT-based formulation has not be found yet.

6 Conclusions

We have introduced the image foresting transformation as a general approach to the design of image processing operators. There are some important characteristics of IFT that support this claim: (a) It is general enough to enable efficient solutions to a broad class of applications; (b) It hides the implementation details of the underlying graph algorithms into a single function, leaving the designer free to concentrate on the function arguments that are specific to his(her) problem; (c) All image operators based on IFT can benefit from the most efficient graph-search algorithm available today, and any changes in such algorithm do not affect the other parts of the application that are specific to the problem.

The central idea of IFT is to translate an image processing problem into a shortest-path forest problem. We have described the IFT for region growing, edge detection, Euclidean distance transform, geodesic distance computation, and watershed transformation. All the operators are efficiently computed using the same IFT algorithm based on the same set of parameters by changing only their meaning and values. We have presented a new interactive image segmentation paradigm based on the region growing operator and an efficient solution to compute the exact Euclidean distance transform.

Currently, we are pursuing the development of interactive image segmentation algorithms based on IFT and investigating the iterative use of IFT for multiscale image segmentation problems. Our goal is to design efficient indexing techniques for content-based image retrieval.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1993.

[2] S. Beucher and F. Meyer. The morphological approach to segmentation: The watershed transformation. *Mathematical Morphology in Image Processing*, pages 433–481, 1993.

[3] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, New York, NY, 1991.

[4] O. Cuisenaire and B. Macq. Fast Euclidean distance transformation by propagation using multiple neighborhoods. *Computer Vision and Image Understanding*, 76(2):163–172, Nov 1999.

[5] R.B. Dial. Shortest-path forest with topological ordering. *Communications of the ACM*, 12(11):632–633, Nov 1969.

[6] A.X. Falcão and J.K. Udupa. Segmentation of 3D objects using live-wire. In *Medical Imaging 1997*, volume 3034, pages 228–239, Newport Beach, CA, Feb 1997. SPIE.

[7] A.X. Falcão, J.K. Udupa, and F.K. Miyazawa. An ultra-fast user-steered image segmentation paradigm: Live-wire-on-the-fly. In *Medical Imaging 1997*, volume 3661, pages 184–191, San Diego, CA, Feb 1999. SPIE.

[8] A.X. Falcão, J.K. Udupa, S. Samarasekera, S. Sharma, B.E. Hirsch, and R.A. Lotufo. User-steered image segmentation paradigms: Live-wire and live-lane. *Graphical Models and Image Processing*, 60(4):233–260, Jul 1998.

[9] K. Fukunaga, H. Murata, T. Asano, and M. Izumi. Image registration using an image graph and its application to map matching. *IEE Proceedings-E Computers and Digital Techniques*, 138(2):79–84, Mar 1991.

[10] G. Gerig, O. Kübler, R. Kikinis, and F. Jolesz. Nonlinear anisotropic filtering of MRI data. *IEEE Transactions on Medical Imaging*, 11:221–232, 1992.

[11] B.E. Hirsch, J.K. Udupa, and S. Samarasekera. A new method of studying joint kinematics from 3D reconstructions of MRI data. *Journal of the American Podiatric Medical Association*, 86(1):4–15, 1996.

[12] C. Lantujoul and F. Maisonnneuve. Geodesic methods in image analysis. *Pattern Recognition*, 17(2):177–187, 1984.

[13] R.A. Lotufo and A.X. Falcão. The ordered queue and the optimality of the watershed approaches. In *International Symposium on Mathematical Morphology' 2000*, Palo Alto, CA, Jun 2000. submitted.

[14] J. Maintz, P. van den Elsen, and M. Viergever. Comparison of edge-based and ridge-based registration of CT and MR brain images. *Medical Image Analysis*, 1:151–161, 1996.

[15] D.K. Mitrakos and A.G. Constantinides. Graph theoretic approach to composite source model estimation for image coding. *IEE Proceedings-F Computers and Digital Techniques*, 131(1):71–79, 1984.

[16] O.J. Morris, M.D.J. Lee, and A.G. Constantinides. Graph theory for image analysis - An approach based on the shortest spanning tree. *IEE Proceedings-F Computers and Digital Techniques*, 133(2):146–152, Apr 1986.

[17] K. Park, I.D. Yun, and S.U. Lee. Color image retrieval using hybrid graph representation. *Image and Vision Computing*, 17:465–474, 1999.

[18] S. Raya and J.K. Udupa. Shape-based interpolation of multidimensional objects. *IEEE Transactions on Medical Imaging*, 9:32–42, 1990.

[19] R.C. Rhoad, J.J. Klimkiewicz, G.R. Williams, S.B. Kesmodel, J.K. Udupa, B. Kneeland, and J.P. Iannotti. A new in vivo technique for 3D shoulder kinematics analysis. *Skeletal Radiology*, 27:92–97, 1998.

[20] Y.M. Sharaiha and N. Christofides. A graph theoretical approach to distance transformations. *Pattern Recognition Letters*, 15:1035–1041, Oct 1994.

[21] A. Shokoufandeh and S. Dickinson. Applications of bipartite matching to problems in object recognition. In *IEEE Workshop on Graph Algorithms and Computer Vision at International Conference on Computer Vision (ICCV'99)*, Corfu, Greece, Sep 1999. http://www.cs.cornell.edu/ICCV-graph-workshop/.

[22] E. Stindel, J.K. Udupa, B.E. Hirsch, D. Odhner, and C. Couture. 3D MR image analysis of the morphology of the rear foot: Application to classification of bones. *Computerized Medical Imaging and Graphics*, 23:75–83, 1999.

[23] D.R. Thedens, D.J. Skorton, and S.R. Fleagle. Methods of graph searching for border detection in image sequences with applications to cardiac magnetic ressonance imaging. *IEEE Transactions on Medical Imaging*, 14(1):42–55, Mar 1995.

[24] J.K. Udupa, B.E. Hirsch, S. Samarasekera, H. Hillstrom, G. Bauer, and B. Kneeland. Analysis of in vivo 3D internal kinematics of the joints of the foot. *IEEE Transactions on Biomedical Engineering*, 45:1387–1396, 1998.

[25] Z. Wu and R. Leahy. An optimal graph theoretical approach to data clustering - Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, Nov 1993.

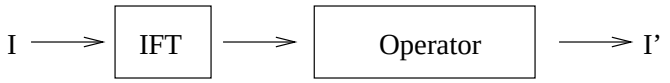

Figure 1: A scheme of a generic image operator based on the image foresting transformation (IFT).

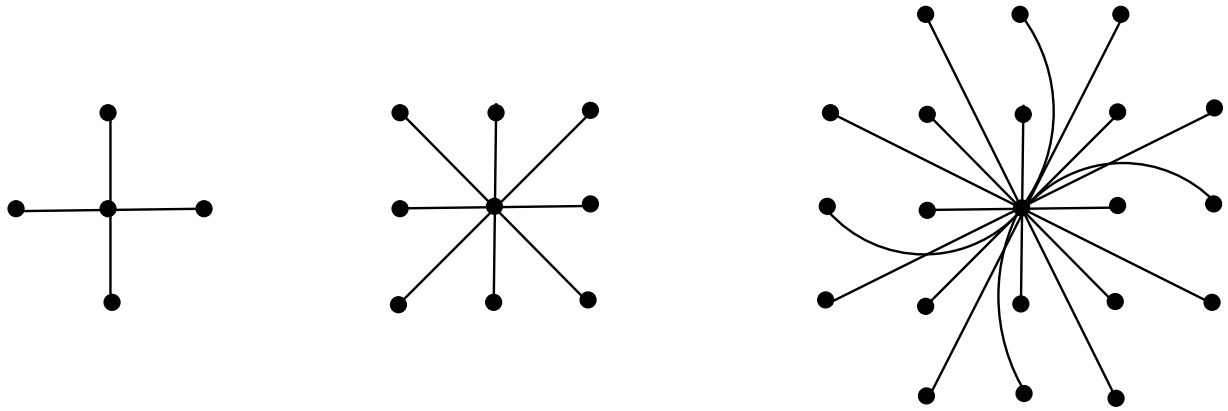

Figure 2: Three examples of adjacency relations for 2-dimensional images. (a) $\rho = 4$, (b) $\rho = 8$, and (c) $\rho = 20$.

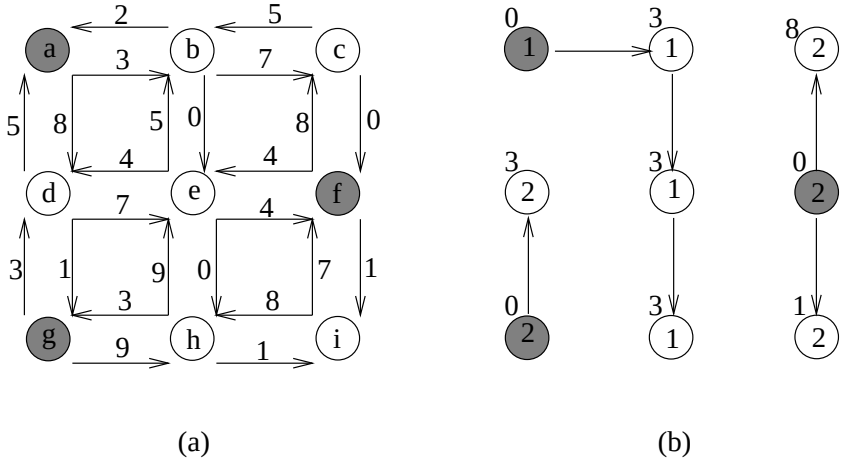

Figure 3: A 2-dimensional example. (a) The graph $\mathbf{G}$ for $\rho = 4$ and (b) a forest in $\mathbf{G}$ created from the roots in $\mathcal{R} = \{\{a\}, \{f, g\}\}$ using **pf** as in Equation 1.

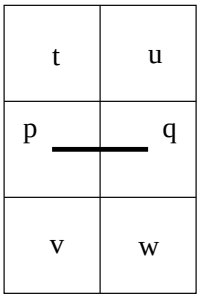

Figure 4: A local neighborhood of two 4-adjacent pixels p and q that we use to compute $\mathbf{w}(p,q)$ based on Equation 5.

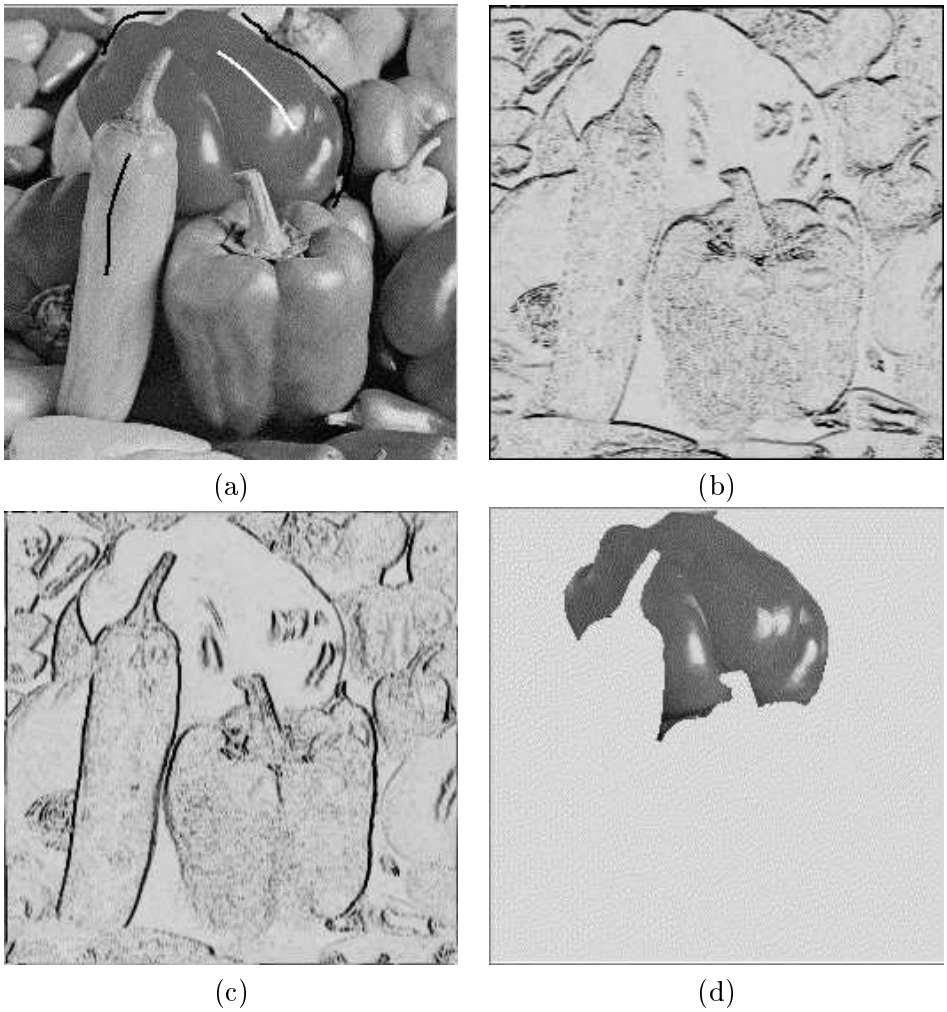

Figure 5: Region growing by IFT. (a) Gray-scale display of a $2D3B$ colored image. The roots selected inside the big dark pepper on the top are represented by white lines, while black lines represent the roots selected outside. (b) and (c) The weight function at the horizontal and vertical arcs, respectively, by using $\rho = 4$, and $\mathbf{w}$ as in Equation 5. Here brightness is inversely proportional to the weight assigned to each arc. (d) The result of extracting the big dark pepper using $\mathbf{pf}$ as in Equation 7.

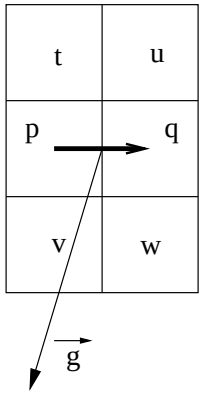

Figure 6: A local neighborhood of two 4-adjacent pixels p and q that we use to compute $\mathbf{w}(p,q)$ in Equation 8, where $\vec{g}(p,q)$ points to the right side of (p,q).

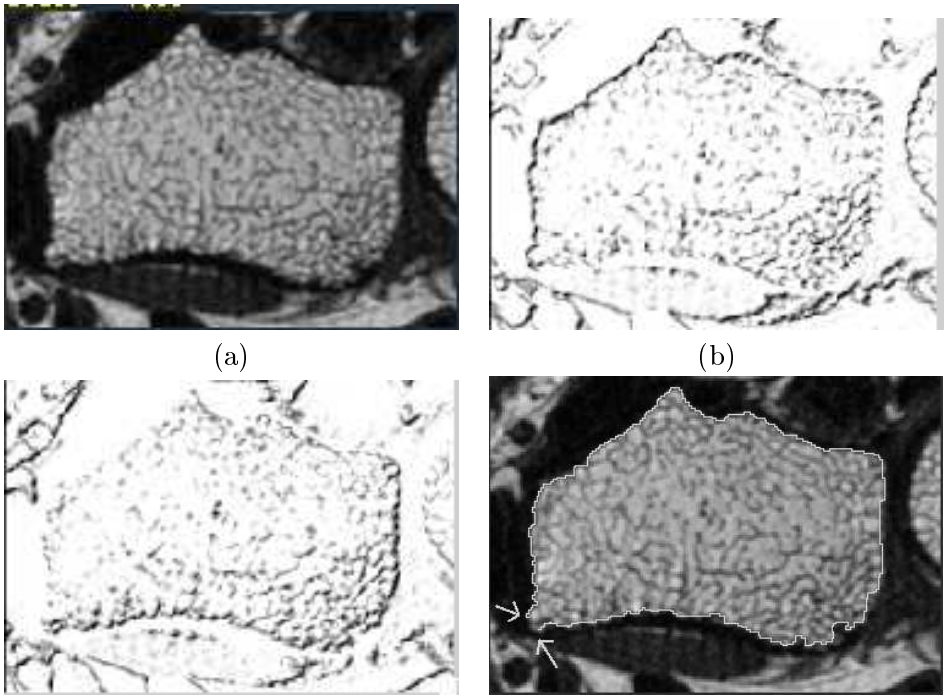

Figure 7: (a) A $2D1B$ MR image of a wrist. (b) and (c) The average between the weight values assigned to up- and right-oriented arcs, and to down- and left-oriented arcs, respectively, for $\rho = 4$ and $\mathbf{w}$ as in Equation 8. Here brightness is directly proportional to weight. (d) A clockwise oriented edge computed using $\mathbf{pf}(r,p)$ as in Equation 1 where r is a root and p a pixel both selected on the boundary of the wrist.

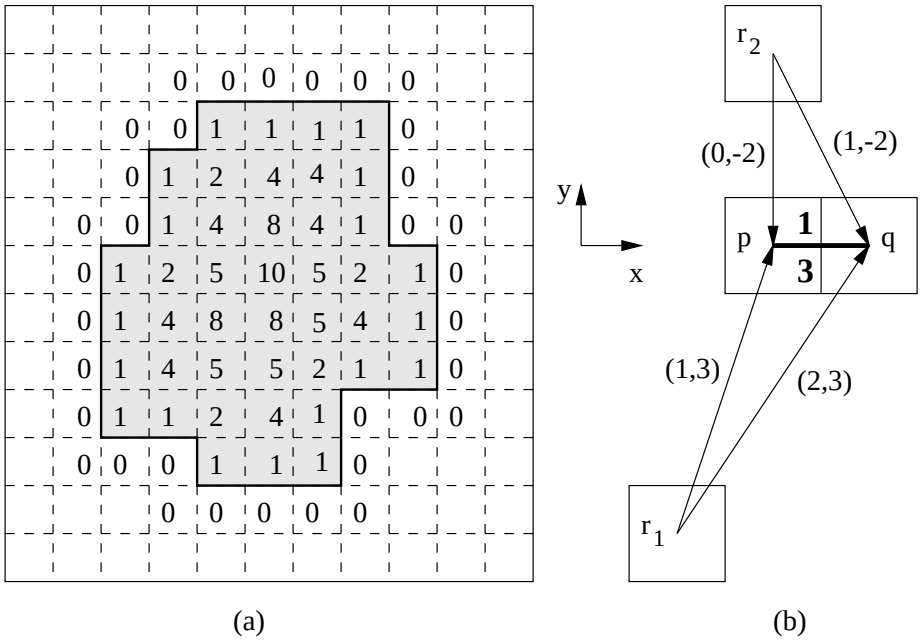

Figure 8: (a) The Euclidean distance transform on a binary image, where the values assigned to each pixel are the squared distance values. (b) The weight assigned to an arc (p, q) may be different for different background pixels.

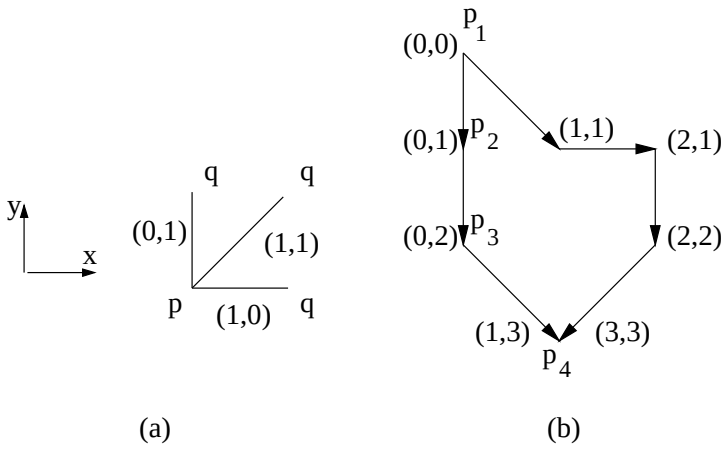

Figure 9: (a) A vectorial representation for $\mathbf{w}(p, q)$ as one of three increment vectors $(dx(p, q), dy(p, q))$: $(0, 1)$, $(1, 0)$, and $(1, 1)$. (b) Two paths from a root p_1 to a node p_4 with the accumulated displacements to each node. Note that, the path $< p_1, p_2, p_3, p_4 >$ with $\mathbf{pf}(p_1, p_4) = 10$ is the one with the correct Euclidean distance.

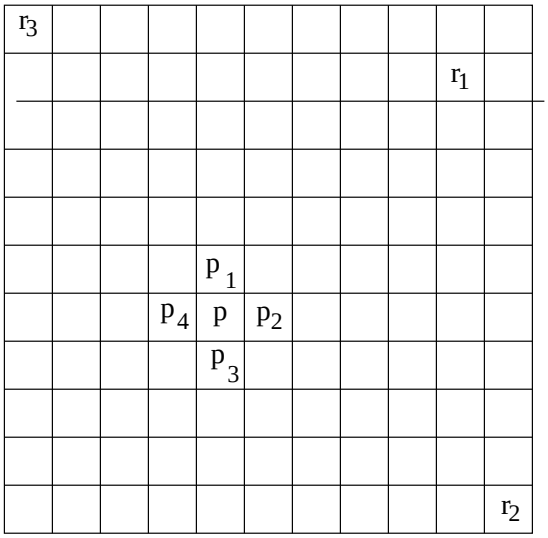

Figure 10: A situation where the Euclidean distance transform might fail if $\rho = 4$.

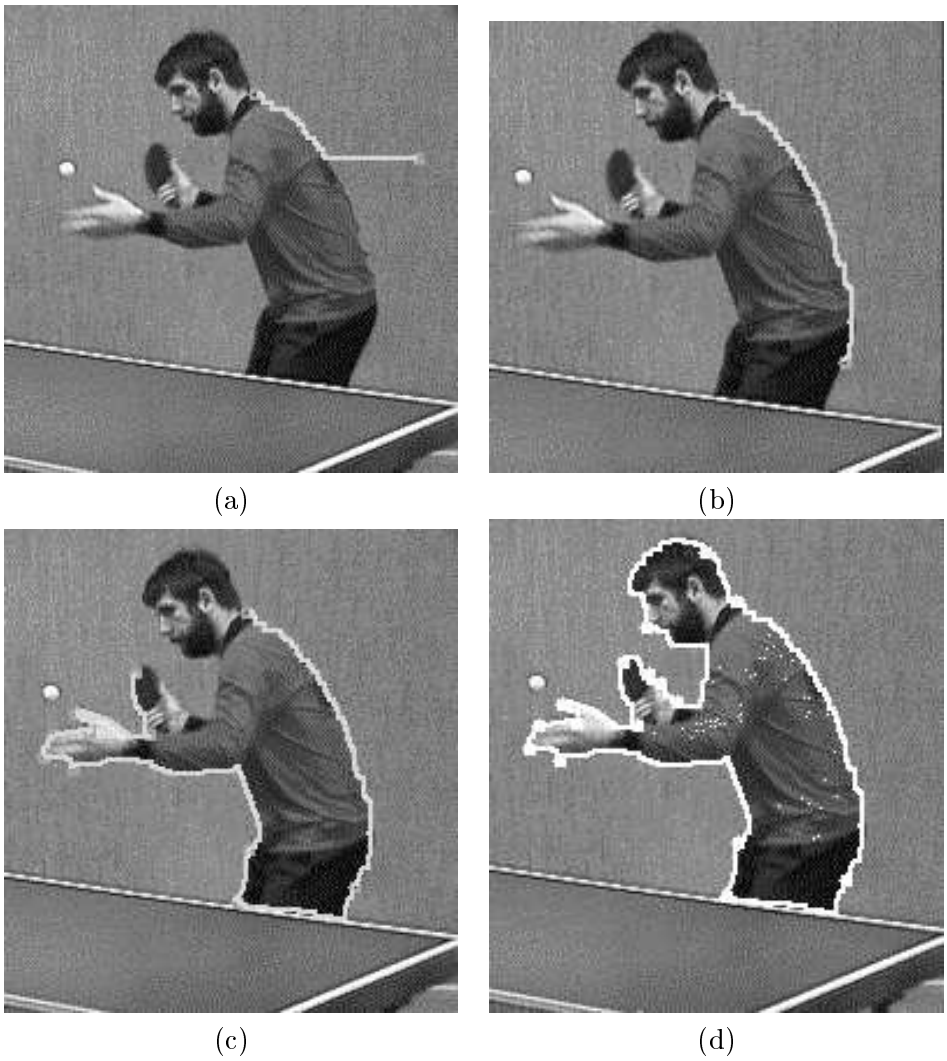

Figure 11: A digital video frame where a tennis player is the object of interest. (a) An optimum edge from an initial point selected on the boundary to the current position of the cursor. (b) An object edge is found by placing the cursor close to the boundary. (c) The delineation process after some points selected on the boundary. (d) The result of segmentation.

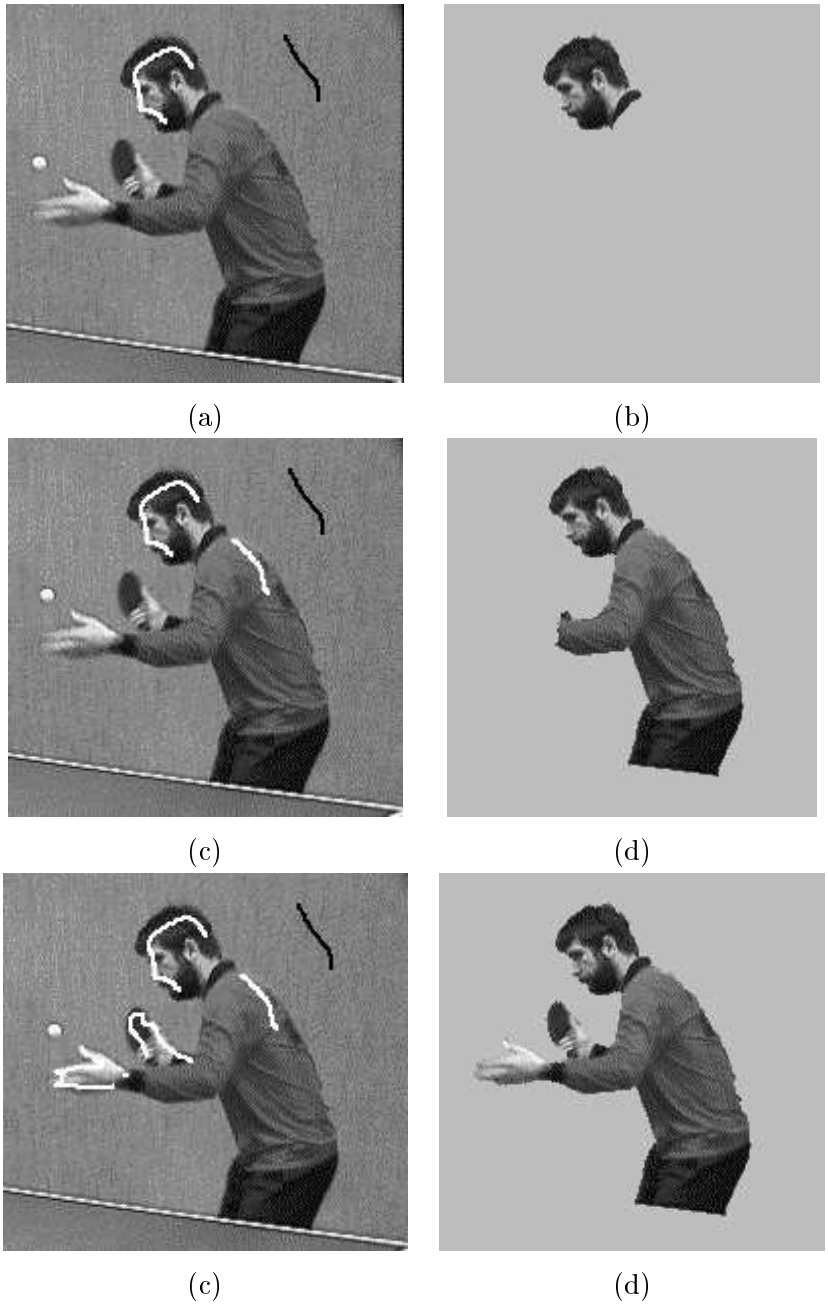

Figure 12: A digital video frame where a tennis player is the object of interest. (a) White line representing roots selected inside the object and black line representing roots selected in the background. (b) Result of segmentation from roots selected in (a). (c) New roots are added inside the object. (d) Result of segmentation from roots selected in (c). (d) Final set of roots selected inside and outside the object. (e) Result of segmentation from roots selected in (d).

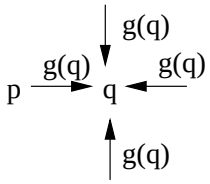

Figure 13: Example for $\rho = 4$ of the arc weight assignment to build a watershed operator using the IFT framework. The weight assigned to each arc (p, q) is the morphological gradient $g(q)$ computed at q. That is, all incoming arcs to a pixel q have the same weight value $g(q)$.

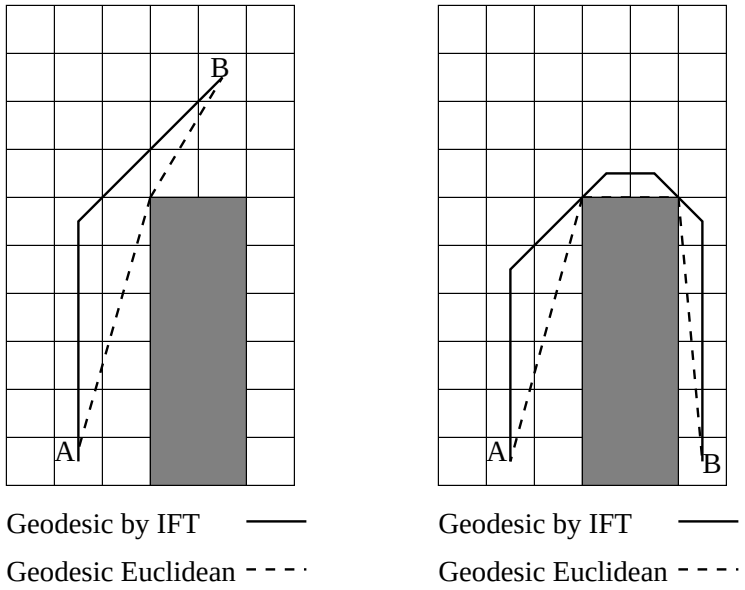

Figure 14: Two examples of geodesic path from A to B in a binary image.