GIC Technical Report n. (GIC-RR-2009-11-12) : Technical report on mathematical and geometric tools for shape recognition using skeletons

Andoni Beristain Iraola (beristainandoni@yahoo.es) and Manuel Graña

November 2009

Abstract

This document introduces several general knowledge required for the comprehension of shape recognition using skeletons. Shape skeletons, and specifically the Medial Axis Transform (MAT) is a shape representation method which permits a compact and accurate description of a shape. If the Distance Transform function value is also provide for each skeleton pixel, as it occurs in the case of the MAT, the shape reconstruction from the skeleton is also possible. Skeleton matching algorithms usually involve an initial abstraction of the skeleton into an Attributed Skeletal Graph (ASG), and then graph matching techniques are used in order to obtain the maximum isomorphism or homomorphism between a sample shape and a set of template graphs corresponding to the shape classes to be recognized.

Contents

1 Introduction

This document presents several miscellaneous knowledge related to shape recognition based in the Medial Axis and skeletons. Section 2 describes the Distance Transform function. Section 3 describes a shape boundary curve sampling method called Discrete Curve Evolution. Section 4 describes the Voronoi Tessellation. Section 5 introduces the different Graph Matching problems and the related algorithms. And finally, Section 6 introduces and reviews Shock Graphs. Most of the sections are introductory, but references to more advanced papers on each topic are provided.

This information has been gathered during the PhD Thesis work carried out by Andoni Beristain, and it is an extended version of one of the Appendixes in his PhD Thesis Report.

2 Distance Transform function (DT)

A distance transform, also known as distance transformation, distance map or distance field, is a representation of a digital image. The choice of the term depends on the point of view on the object in question: whether the initial image is transformed into another representation, or it is simply endowed with an additional map or field. The map supplies each pixel of the image with the distance to the nearest obstacle pixel. A most common type obstacle pixel is a boundary pixel in a binary image. Usually the transform/map is qualified with the chosen metric. For example, one may speak of Manhattan distance transform, if the underlying metric is Manhattan distance. Common metrics are:

- Manhattan distance, City block distance or Taxicab geometry (d_1). This distance is defined as $d(p,q)_1 = \|p,q\|_1 = \sum_{i=1}^{n} |p_i - q_i|$, where $p = (p_1, ..., p_n)$and $q = (q_1, ..., q_n)$are vectors of n dimensions, and $|\cdot|$ is the absolute value.
- Chessboard distance, Chebyshev distance or Tchebychev distance (d_∞). It is defined as $d_{Chebyshev}(p,q) = \max_i (|p_i - q_i|)$
- Chamfer distance, or weighted distance, with weights $(a,b)(d_{a,b})$. Let $r, s \in \mathbb{Z}^2$and ρ a sequence of king's moves from p to q. m = number of isothetic moves; n =number of diagonal moves Then it is defined as $d_{a,b}(p,q) = \min_\rho (l_{a,b}(\rho))$ with $l_{a,b}(\rho) = ma + nb$.
- Euclidean distance. It is defined as $\|p,q\| = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$.

Given a binary image, the distance transform function value of an image point $x \in \mathbb{R}^2$, denoted as $DT(x)$is the minimum distance of x to any image background point. Formally, let a binary image $I : \mathbb{R}^2 \to \{0,1\}$where 0 corresponds to the background B and 1 to the foreground F (i.e., the shape), and $I = B \cup F$, and a distance measure $|.|_d$. Then the distance transform function is a function $DT(x)$ so that,

$$DT(x) = \min (|x - y|_d) \ \ s.t.\ y \in B \tag{1}$$

Initial Distance Transform computation algorithms made use of distance metrics alternative to the Euclidean in order to improve the efficiency, e.g., [1, 2, 3, 4]. These distance measures benefit from the local properties of their metrics to avoid the global minimization of the distance independently for each pixel. But the euclidean distance does not have local properties. The main problem of this alternative metrics is that they are usually less accurate (according to [5] there is distance measurement error between 40 and 5 percentage with the Euclidean distance) but also orientation dependent. The most rotational invariant metric is the Euclidean. Fortunately, actual algorithms like like [6, 7, 8] can compute

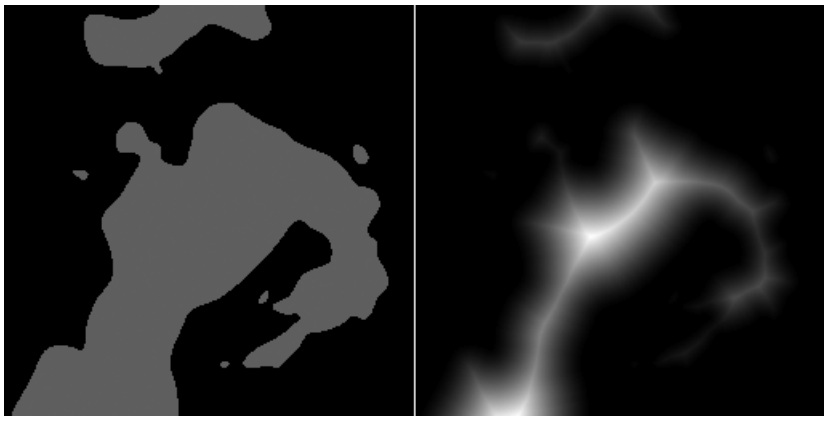

Figure 1: Example of graphical representation of the Distance Transform function. The left image represents several connected components in gray, and the background in black. And the image on the right shows the distance transform value for each pixel in the left image. Brighter pixel color represents higher Distance Transform value.

the Euclidean Distance Transform in $O(n)$. For a review on Euclidean Distance Transform algorithms the reader is referred to [9, 10, 11]. A graphical representation of this transform can be seen in fig. 1.

Some properties of the Distance Transform Function:

- Its value is positive definite, that is $d(p,q) \geq 0$, and 0 only when $p = q$, $\forall p, q \in S$, where S represents the foreground of connected component set in the image, and it is limited by the width and height of the image.
- It is symmetric, $d(p,q) = d(q,p)$, $\forall p, q \in S$.
- Triangular $d(p,q) \leq d(p,r) + d(r,q), \forall p, q \in S$

3 Discrete Curve Evolution (DCE)

Discrete Curve Evolution [12], DCE henceforth, is a shape boundary curve downsampling method that keeps the most meaningful geometrical information for shape recognition [13, 14]. Boundary curves of digital binary images usually include some degree of noise due to the digitization and segmentation errors. And the set of boundary points usually includes redundant points, or at least many points which provide very low information for the shape description, i.e. they have low importance or salience. The main idea of the DCE procedure is to downsample iteratively the shape boundary curves' point set, removing the point with less contribution to the global shape at each step of the iteration, therefore reducing the boundary curve complexity while keeping the most important information of the shape.

Any digital curve can be described as a polygon, with the enough number of vertices, and the main idea of the DCE procedure consists of removing one edge of that polygon at each step of the procedure by replacing the two consecutive segments connected to the point with the global minimum relevance value v_{min} by one connecting the two points left after the removal of that v_{min}.

The salience of each vertex is given by the relevance value $K(S_i, S_{i+1})$, which computes the salience of the polygon vertex incident to the consecutive polygon segments S_i and S_{i+1}. It is defined as:

$$K(S_i, S_{i+1}) = \frac{\beta(S_i, S_{i+1}) l(S_i) l(S_{i+1})}{l(S_i) + l(S_{i+1})}, \tag{2}$$

where $\beta(S_i, S_{i+1})$ is the turn angle at the common vertex of segments S_i and S_{i+1} and l is the length function normalized with respect to the total length of a polygonal curve C. The value $K(S_i, S_{i+1})$ is directly proportional to the contribution of the arc $S_i \cup S_{i+1}$ to the shape. The cost function $K(S_i, S_{i+1})$ is monotonically increasing with respect to the relative lengths and the total curvature of segments S_i and S_{i+1}

[12]Let $D_m = s_0, ...s_{m-1}$be a decomposition of a digital curve C into consecutive digital line segments. The algorithm that computes the decompositions D_k for each stage of the discrete curve evolution $k > 3$ until $D_{k-1}$is convex is the following:

This algorithm is guaranteed to terminate because the number of initial segments is finite and one segment is removed at each step of the algorithm. It is also proved in [12] that the DCE procedure converges into a convex polygon. Although the original algorithm ends when the convexity of the DCE polygon is achieved, other ending criteria is possible, e.g. defining a constant number of iterations or final segment number. In [14] a more sophisticated criteria is employed based on a threshold value for the difference between the original shape and the DCE approximation.

Properties of the Discrete Curve Evolution procedure:

Algorithm 1 Discrete Curve Evolution Algorithm (D_m)

$k = m$;

Do

Find in (D_k) a pair s_i , $s_{(i+1)mod(k)}$ such that $K(S_i, S_{i+1})$ is minimal

$D_{k+1} = D_k$ with segments s_i,s_{i+1} replaced by s' joining the endpoints of $s_i U s_{i+1}$;

$k = k - 1$;

until D_{k-1} is convex;

- It leads to the simplification of shape complexity, in analogy to evolutions guided by diffusion equations.
- There are no blurring (i.e., shape rounding) effects and no dislocation of relevant features, because vertices do not change their position.
- The relevance value $K(S_i, S_{i+1})$ is stable under deformations on the shape boundary curves, because noise elimination takes place in the early stages of the evolution.
- It allows to find line segments in geometrical objects even under noisy images, due to the relevance order of the repeated processes of digital linearization.

4 Voronoi Tessellation in $\mathbb{R}^2$

A Voronoi Tessellation, also called Voronoi diagram, Dirichlet diagrams or Thiessen polygons, is a partition of the space into convex regions called Voronoi polygons, each around a generator or Voronoi site, (belonging to a finite set of points in the space with at least two points), so that the pixels in the Voronoi polygon o an specific Voronoi site are closer to it than to any other Voronoi site. This partition is illustrated in figure 2.

The first works presenting Voronoi diagrams and many related issues are those of Peter Gustav Lejeune Dirichlet (1805-1859) [15] and Georgy Fedoseevich Voronoy (Georges Voronoi) (1868-1908) [16, 17, 18]. Dirichlet focused his work in two and three dimensions, while Voronoi generalized his results to arbitrary high dimensions. Their concern was with the distribution of points with integer coordinates that give the minimum of the values of a given quadratic form. In that context, they considered the set of points regularly placed in the m-dimensional space generated by linear combinations of M linearly independent vectors with integer coefficients. This set contains infinitely many points, and the Voronoi diagram generated by this set of points gives the partition of the space into mutually congruent polyhedra. One of the first and most important applications of Voronoi diagrams occurred in meteorology when Thiessen (1911) [19] used Voronoi regions as an aid to computing more accurate estimates of regional rainfall averages. Whitney (1929) [20] refers to the procedure as 'Thiessen's method' and since then the term *Thiessen polygon* has remained a popular one in two-dimensional applications in meteorology, geography and related social science disciplines.

The Voronoi Tessellation can be computed in $O(n \log n)$[21], but an optimal lower bound in $O(n)$ is proved to be theoretically feasible in [22], with n being the number of $v_i \in V_{sites}$. For convex polygons the computational cost is $O(n)$, as proved by Aggarwal in [23]. Although this tessellation can be generalized to n-dimensional spaces, this section is focused in the bi-dimensional space. Next, the Voronoi Tessellation is mathematically defined, as well as its dual, the Delaunay triangulation.

4.1 Planar Ordinary Voronoi Tessellation/Diagram

Planar ordinary Voronoi diagram. Given a set of two or more but a finite number of distinct points in the Euclidean plane, we associate all locations in that space with the closest member(s) of the point set with respect to the Euclidean distance. The result is a tessellation of the plane into a set of the regions associated with members of the point set. We call this tessellation the planar ordinary Voronoi diagram generated by the point set, and the regions constituting the Voronoi diagram ordinary Voronoi polygons. Mathematically,

Let the Voronoi Tessellation V consists of a decomposition of the image plane domain D into convex regions around a set of points, the Voronoi sites $V_{sites} = \{v_1, ..., v_n\}$, and called *Voronoi polygon* $V_{poly}(i)$. A Voronoi polygon $V_{poly}(i)$ around a Voronoi site v_i is defined as:

$$V_{poly}(i) = \left\{ x, v_i, v_j \in \mathbb{R}^2 \mid \|x - v_i\| \leq \|x - v_j\| \, \forall v_j \neq v_i \right\} \tag{3}$$

Then, the Voronoi Tessellation is mathematically defined as,

$$V = \bigcup_{i=1}^{n} V_{poly}(i) = \{V_{poly}(1), ..., V_{poly}(n)\} \tag{4}$$

Each Voronoi polygon is bounded by several Voronoi segments and vertices.

A *Voronoi segment* is the locus of the intersection of exactly two and no more different Voronoi polygons, and it is therefore determined only by two Voronoi sites. Consequently the Voronoi segment shared by the Voronoi sites v_i and v_j, denoted as s_{ij} is defined by the next equation:

$$s_{ij} = V_{poly}(i) \cap V_{poly}(j) \tag{5}$$

or alternatively,

$$s_{ij} = \left\{ x \in \mathbb{R}^2 \mid \|x - v_i\| = \|x - v_j\| \wedge \forall k \neq i \neq j, \|x - v_k\| > \|x - v_i\| \right\} \tag{6}$$

A *Voronoi vertex* is the intersection of three or more Voronoi polygons. The set of Voronoi vertices is defined as:

$$T = \left\{ x \in \mathbb{R}^2 \middle| W \subset V_{sites} \wedge ||W| \geq 3| \; s.t. \; x \in \bigcap_{ij, \in W} s_{ij} \right\} \tag{7}$$

In fig. 2 a graphical representation of a simple planar Voronoi Tessellation is shown. These definitions show that there must be at least two points in order to compute the Voronoi Tessellation.

4.2 Delaunay Triangulation

A Voronoi diagram has its 'dual tessellation', called a Delaunay triangulation, which is represented in fig. 3. We consider a Voronoi diagram in the Euclidean plane, and assume that generator points of the Voronoi diagram are not on the same line, and also that there are at least three Voronoi sites, but they are finite $3 \leq |V_{sites}| < \infty$, otherwise the Delaunay triangulation is not possible.

To obtain the Delaunay Triangulation from the Voronoi Tessellation, first we choose a Voronoi edge in a Voronoi diagram (the heavy broken line in fig 3(a)). This Voronoi edge is shared by two Voronoi polygons. We join the generator points (the filled circles) of these Voronoi polygons by a line segment (the heavy solid line in fig 3(a)). We carry out this line generation with respect to all Voronoi edges in the Voronoi diagram. As a result, we obtain a second tessellation of the convex hull of the generator points (the solid lines in 3(a)). Another tessellation is shown in fig. 3(b). These two tessellations are different in that the tessellation in panel (a) consists of only triangles, whereas the tessellation in panel (b) contains a quadrangle. We call the former tessellation a

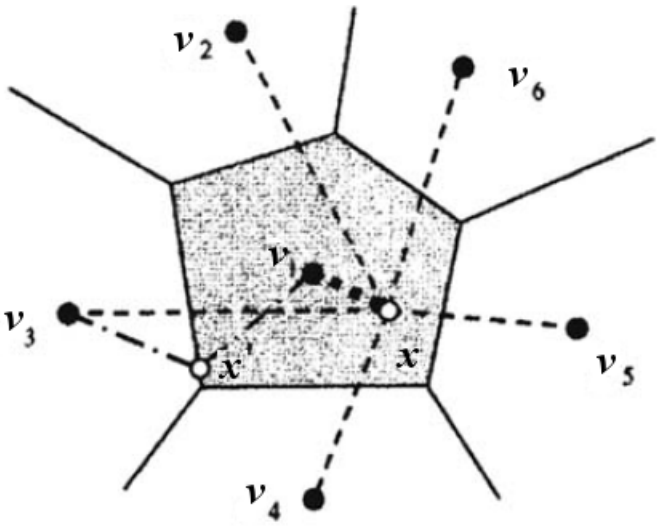

Figure 2: Graphical representation of a simple planar Voronoi Tessellation. v correspond to Voronoi sites, while x correspond the ordinary points. $xı$ belongs to s_{13}.

Delaunay triangulation. The latter is not a triangulation, but since it is to be triangulated, we call it a *Delaunay pretriangulation*.

A Delaunay pretriangulation has polygons having four or more vertices. These non-triangular polygons are decomposed into triangles by non-intersecting line segments joining the vertices, For example, the quadrangle in fig. 3(b) with vertices $p_{i1}, p_{i2}, p_{i3}, p_{i4}$ can be partitioned into two triangles by the segment $\overline{p_{i1}p_{i3}}$ or $\overline{p_{i2}p_{i4}}$ obtaining the triangulations in the bottom left and right pictures, therefore this procedure is not unique and both are valid triangulations. This kind of tessellation is usually called Delaunay Triangulation too, but in some cases is called degenerate Delaunay triangulation to distinguish it from the above Delaunay triangulation. If the noncollinearity assumption and $3 \leq |V_{sites}| < \infty$ are not satisfied, this procedure does not produce triangles but line segments, or triangles which degenerate into line segments.

Most of the Delaunay triangulation algorithms can perform in $O(n \cdot \log(n))$ time, but in [24] a method is proposed to compute it in expected $O(n)$ time. And the conversion between the Voronoi Tessellation and the Delaunay triangulation can be performed in lineal time ($O(n)$).

Delaunay edge: Let V_{sites} be a finite set of points in a sub-domain Ω^2 of the bi-dimensional space $\mathbb{R}^2$. Two points v_i and v_j are connected by a Delaunay edge e if and only if there exists a location $x \in \Omega^2$ which is equally close to v_i and v_j and closer to v_i, v_j than to any other $v_k \in V_{sites}$. The location x is the center of an circle which passes through the points v_i, v_j and which contains no other points v_k of V_{sites}.

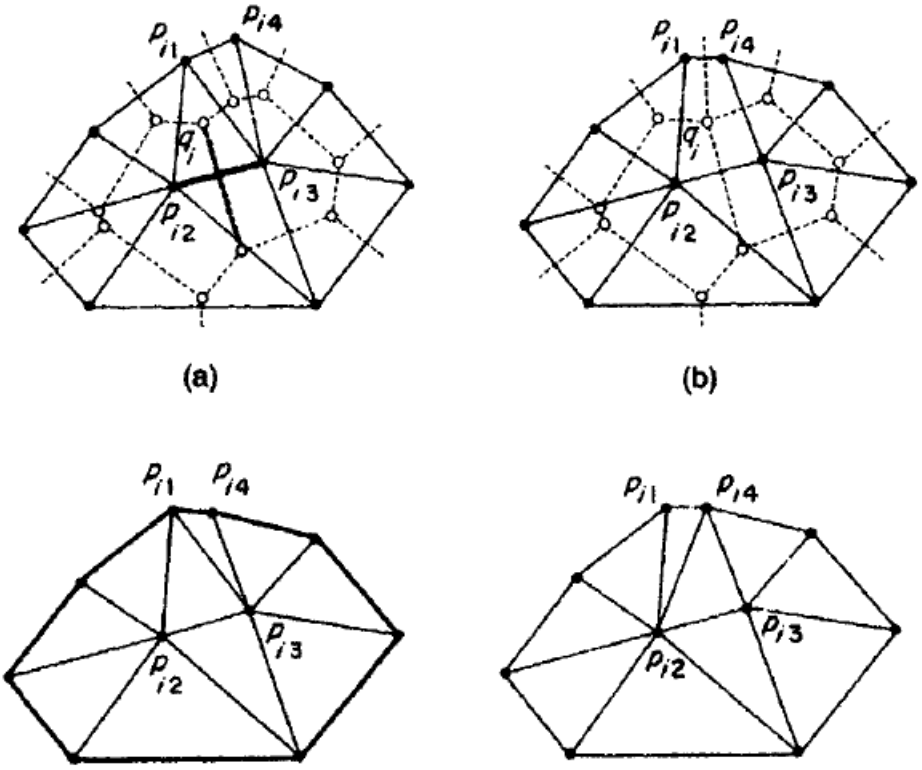

Figure 3: Delaunay triangulation example. Filled circles correspond to Voronoi sites, empty circles correspond to Voronoi vertices, dotted lines correspond to the Voronoi polygons and filled lines correspond to the delaunay triangulation. Figure a shows a complete triangulation, while b is a pretriangulation and the figures below are two possible final Delaunay triangulations computed from b.

$$e_{ij} = \exists x \in \Omega^2 \wedge \|x - v_i\| = \|x - v_j\| \wedge \forall k \neq i, j \ \ \|x - v_i\| < \|x - v_k\| \qquad (8)$$

Delaunay triangle: Let V_{sites} be a finite set of points in a sub-domain Ω^2 of the bi-dimensional space $\mathbb{R}^2$. Three non-collinear points v_i, v_j, v_k form a Delaunay triangle t if and only if there exists a location $x \in \Omega^2$ which is equally close to v_i, v_j and v_k and closer to v_i, v_j and v_k than to any other $v_m \in V_{sites}$. The location x is the center of only one circle (the circumcircle of t) which passes through the points v_i, v_j, v_k and which contains no other points $v_m \in V_{sites}$.

$$t_{ijk} = \exists x \in \Omega^2 \wedge \|x - v_i\| = \|x - v_j\| = \|x - v_k\| \wedge \forall m \neq i, j, k \ \ \|x - v_i\| < \|x - v_m\| \qquad (9)$$

5 Graph Matching

Most of the information contained in this section has been summarized from chapters 2 and 6 of the thesis work by Bengoetxea [25]. For more detailed explanations the user is referred to that work.

5.1 Introduction

Let $G(V, E)$ be a *graph* where V is the set of vertices or nodes and $E \rightarrow V \times V$ (also defined as $E[V]^2$ in the literature) is the set of edges (also known as arcs, links or lines). The order (or size) of a graph G is defined as the number of vertices of G and it is represented as $|V|$ and the number of edges as$|E|$, or also $|G|$ and $\|G\|$. If two vertices in G, say$u, v \in V$, are connected by an edge $e \in E$, this is denoted by $e = (u, v)$ and the two vertices are said to be *adjacent* or neighbors. Edges are said to be *undirected* when they have no direction, and a graph G containing only such types of graphs is called undirected. When all edges have directions and therefore (u, v) and (v, u) can be distinguished, the graph is said to be *directed*. Usually, the term *arc* is used when the graph is directed, and the term edge is used when it is undirected. In addition, a directed graph $G = (V, E)$ is called *complete* when there is always an edge$(u, u\prime) \in E = V \times V$ between any two *vertices* $u, u\prime$ in the graph. A graph $G\prime(V\prime, E\prime)$ is called a *subgraph* of $G(V, E)$ if $V\prime \subseteq V$ and $E\prime \subseteq E$.

Graph vertices and edges can also contain information. When this information is a simple label (i.e. a name or number) the graph is called *labeled graph*. Other times, vertices and edges contain some more information. When only nodes include attributes the graph is said to be *vertex-attributed*, or simply an *attributed graph* and when only the edges are attributed it is called *edge-attributed* or *weighted graph*. When both edge and vertex attributes are present, the graph is called an *attributed relational graph*.

A path between any two vertices $u, u\prime \in V$ is a non-empty sequence of k different vertices $< v_0, v_1, ..., v_k >$ where $u = v_0, u\prime = v_k$ and $(v_{i-1}, v_i) \in E, i = 1, 2, ..., k$. Finally, a graph G is said to be *acyclic* when there are no cycles between its edges, independently of whether the graph G is directed or not. An example of a graphical representation of a graph is shown in fig. 4

5.2 Isomorphism and Homomorphism Graph Matching

The graph matching problem, applied to recognition, consists of obtaining the minimum distance between a sample graph G_s and a set of template graphs $G_T = \left\{G_{T_0}, ..., G_{T_p}\right\}$ corresponding to the classes, i.e. the *similarity* value between them. The similarity or best correspondence of a graph matching problem is defined as the optimum of some objective function which measures the similarity between matched vertices and edges. This objective function is also called fitness function. This similarity between two graphs is usually asymmetric in the way that $d(G_1, G_2) \neq d(G_2, G_1)$. In those cases the maximum or average distance is used. The fitness function can consider several information sources:

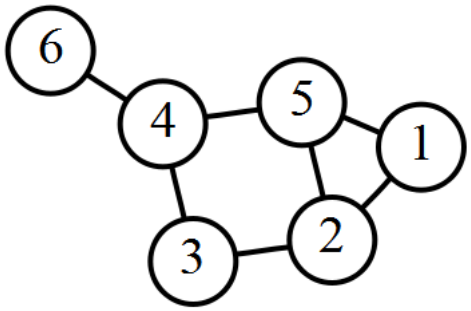

Figure 4: Example of non directed cyclic labeled graph. The circles represent the nodes and the lines represent the links. The numbers in the nodes correspond to their labels.

it can use only the information of the correctly matched vertices, or include the similarity between the labels in the matched vertices too; it can discard graph edge information of those matchings involving n node in the sample to the same node in the model; and it can also use a function based in distribution divergence, which requires the creation of a new graph model using probability parameters.

Formally, given two graphs $G_s = (V_s, E_s)$ and $G_{T_i} = (V_{T_i}, E_{T_i})$, with $|V_s| = |V_{T_i}|$, the graph matching problem is to find a one-to-one mapping $f : V_s \rightarrow V_{T_i}$ such that$(u, v) \in E_s \; iff \; (f(u), f(v)) \in E_{T_i}$. When such a mapping f exists, this is called an *isomorphism*, and G_s is said to be isomorphic to G_{T_i}. This type of problems is said to be *exact graph matching*.

The term *inexact* applied to some graph matching problems means that it is not possible to find an isomorphism between the two graphs to be matched. This is the case when the number of vertices is different in both the model and data graphs. This may be due to the schematic aspect of the model and the difficulty to segment accurately the image into meaningful entities. Therefore, in these cases no isomorphism can be expected between both graphs, and the graph matching problem does not consist in searching for the exact way of matching vertices of a graph with vertices of the other, but in finding the best matching between them. This leads to a class of problems known as *inexact graph matching*, or *homomorphism*. In that case, the matching procedure aims at finding a non-bijective correspondence between a data graph and a model graph. In an inexact graph matching problem the goal is to find a mapping $f\prime : V_s \rightarrow V_{T_i}$ such that $(u, v) \in E_s \; iff \; (f(u), f(v)) \in E_{T_i}$, assuming that $|V_{T_i}| < |V_s|$. This corresponds to the search for a small graph within a big one. An important sub-type of these problems are sub-graph matching problems, in which we have two graphs $G = (V, E)$ and $G\prime = (V\prime, E\prime)$, where $V\prime \subseteq V$ and $E\prime \subseteq E$, and in this case the aim if to find a mapping $f\prime : V\prime \rightarrow V$ such that $(u, v) \in E\prime$ iff $(f(u), f(v)) \in E$. When such a mapping exists, this is called a *subgraph matching* or *subgraph isomorphism*.

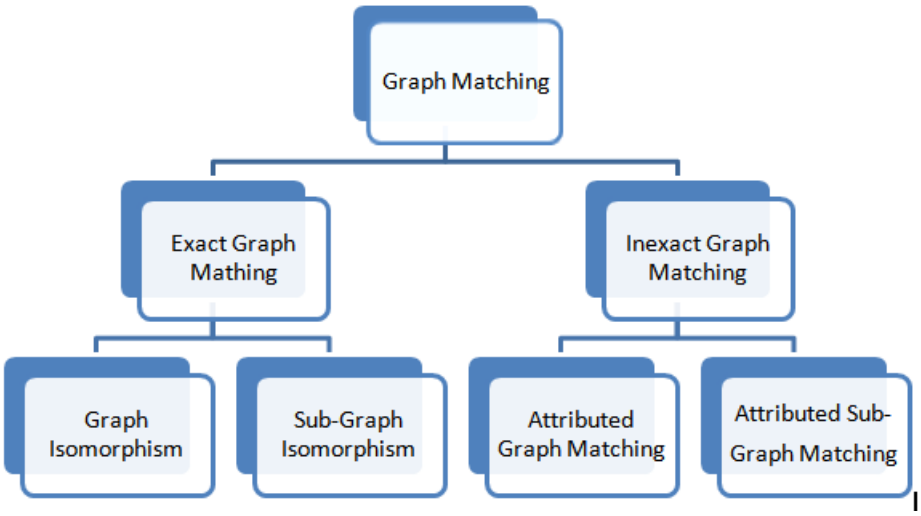

Figure 5: Graph matching type classification

The classification of the different graph matching problems is graphically shown in fig. 5.

5.3 Graph Matching Techniques

A Graph matching algorithm should met this characteristics according to [25]:

1. Its similarity value should increase as the solutions approaches to the optimal, and should avoid local maximas.
2. It should avoid ambiguity, i.e. different samples should have different values.
3. It should take into account only the most relevant attributes of the nodes and links. And in most of cases give different weights to the similarity measure among nodes and among links, because links usually contain more relevant information about the relationship between parts than information about the regions themselves.
4. It should be easy and computationally efficient.

Unfortunately, graph matching problems are known to be NP-complete, with an increasing complexity starting by exact isomorphism, following with subgraph isomorphism and finishing with attributed graph and sub-graph homomorphism matching which are the most complex. Consequently the efficiency of graph matching algorithms is one of the most important goals, including many heuristics, and even trying to transform the graph into a rooted tree when possible, because its lower matching complexity.

Graph Matching using dummy nodes is a special case of the exact graph matching, when $|G_1| \neq |G_2|$. In such cases artificial nodes can be added, so that $|G_1| = |G_2|$, and given values such that the similarity between nodes is very low, to penalize their use in the matching process. It is a one-to-one graph vertex matching, although it is also a homomorphic graph matching, just in the limit with the isomorphic graph matching procedure. It is less computationally complex than conventional many-to-many graph node matching procedures, i.e. graph homomorphic matching, but imposes much more restrictions to the graphs being matched.

Some other graph matching problems allow many-to-many matches, that is, given two graphs$G_{T_i} = (V_{T_i}, E_{T_i})$ and $G_s = (V_s, E_s)$, the problem consists on searching for a homomorphism $f : V_s \rightarrow W$ where $W \in P(V_{T_i}) \setminus \{\emptyset\}$ and $W \subseteq V_{T_i}$. In case of using also dummy vertices, W can take the value $\emptyset$ and therefore $W \in P(V_{T_i})$. This type of graph matching problems are more difficult to solve, as the complexity of the search for the best homomorphism has much more combinations and therefore the search space of the graph matching algorithm is much bigger.

5.3.1 Elastic matching

Elastic graph matching is a graph representation and matching technique that takes into account the possible deformation of objects to be recognized. It usually consists of two steps. First of all both graphs are matched with a rigid grid, and then the grid is deformed, permitting certain flexibility. In shape matching, the second step permits certain deformation, rotation and scale between the template and the sample. In a [26] several algorithms of this type are compared in a practical problem.

5.3.2 Morphological Graph Matching

Morphological graph matching applies hyperplanes or deformable spline-based models to the skeleton of non-rigid discrete objects. The graph is built from the most salient point set in the skeleton. Partial shape recognition is possible interactively defining shape features using a sub-graph matching algorithm.

5.3.3 Graph Edit Distance

The graph edit distance between two graphs is defined as the number of modifications that one has to undertake to arrive from one graph to be the other. The distance between two graphs is defined as the weighted sum of the costs of edit operations (insert, delete, and relabel the vertices and edges) to transform one graph to the other. The fact of applying these concepts and removing vertices or edges in graphs is analyzed in many works, as removal will lead to smaller graphs and therefore the graph matching problem can be reduced in complexity.

5.3.4 Error Correction Graph Matching

In error-correcting graph matching (or error-tolerant graph matching as it is also called) one considers a set of graph edit operations, and defines the edit distance of two graphs G_1 and G_2 as the shortest (or least cost) sequence of edit operations that transform G_1 into G_2 . Error-correcting graph matching is a powerful concept that has various applications in pattern recognition and machine vision, and its application is focused on distorted inputs. It constitutes a different approach very similar to other graph matching techniques. In [27] this topic is addressed and a new distance measure on graphs that does not require any particular edit operations is proposed. This measure is based on the maximal common subgraph of two graphs. A general formulation for error-correcting subgraph isomorphism algorithms is presented in [28] in terms of adjacency graphs, and [29] presents a study on the influence of the definition of fitness functions for error correcting graph matching.which reveals guidelines for defining fitness functions for optimization algorithms in error correcting graph matching. In addition, in [30] an algorithm for error-correcting subgraph isomorphism detection from a set of model graphs to an unknown input graph is introduced.

5.3.5 Other Approaches

Many other techniques have been applied to the graph matching problem. Evolutionary algorithms, and specifically genetic algorithms [31, 32, 33], and the use of techniques based on probability theory [34, 35, 36], including the application of probabilistic relaxation to graph matching [37] and the Expectation Maximization (EM) algorithm [38]. Decision trees have aso been used for graph matching like in [39], Neural Networks [40, 41], clustering techniques [42], etc [43, 44, 45, 46, 47, 30].

5.3.6 Multiple Graph Matching

Sometimes a sample graph G_s has to be compared against a whole database of graphs representing different classes $G_T = \{G_{T_0}, ..., G_{T_p}\}$. In such cases, the goal is to find the model template more similar to the sample graph, i.e. $d(G_s, G_T) = \min_{0 \leq i \leq p} d(G_s, G_{T_i})$, and assign that graph template class $G_{T_i}$to the input sample G_s. And that minimum distance $d(G_s, G_{T_i})$ also gives a measure the difference between the input sample and the assigned class. Comparing this measure to the distance with the second and third most similar graph template classes, also gives a confidence measure of the class assignment.

Obtaining the most similar graph template class initially requires to compute the similarity between the sample graph and every graph template in the database, which requires a high computational cost as the size of the database increases. In order to minimize the cost of this procedure, graph template indexing can be performed over the database avoiding to compute every similarity measure. Graphs in the database are structured as a decision tree, where several steps analyzing different specific characteristics of the input sample graph

discard part of the database at each step. The process ends when only the X most similar graphs in the database remain similar. If $X = 1$ only one graph template and the similarity value is returned, and otherwise a ranking of the X graph is provided in descending order of the similarity. In [48] a attributed graph database indexing is proposed.

6 Shock Graph

6.1 Introduction

A Shock graph is a shape representation method. It is based in the Medial Axis but includes additional information, producing a graph representation of the shape which keeps global and local information of the shape, but also the relationship between parts. The main goals of this representation are:

- Viewpoint independent to start.
- Generic, in the sense that a notion of equivalence classes of qualitatively similar shapes emerges (e.g., hands, houses,...).
- Applicable to natural as well as man-made objects (i.e., amorphous against regular shaped objects).
- Reliably and stably computable.
- Supporting efficient (e.g., polynomial-time) recognition in the presence of occlusion and noise.
- Places special importance on certain boundary segments.

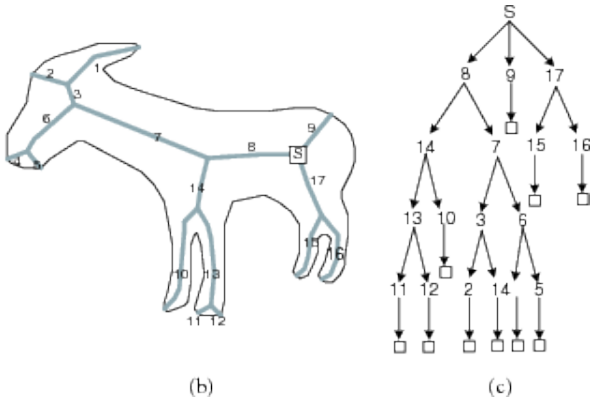

Figure 6: Shock Graph example

6.2 Shapes and shocks

Shocks [49] are entropy satisfying entities, and the locus of shock positions forms the Blum's Medial Axis. The categorization of shocks according to the local variation of the radius function along the medial axis produces a labeled Medial Axis which provides a much richer shape descriptor than an unlabeled skeleton. To illustrate the labeling, imagine traversing a path along the medial axis. At a 1-shock the radius varies monotonically, as is the case for a protrusion. At a

2-shock the radius function achieves a strict local minimum such that the medial axis is disconnected when the shock is removed, e.g., at a neck. At a 3-shock the radius function is constant along an interval, e.g., for a bend with parallel sides. Finally, at a 4-shock the radius function achieves a strict local maximum, as is the case when the evolving curve annihilates into a single point or a seed. A visual representation of each category can be seen in figure 7.

Formally, this medial axis labeling based in shock categories is defined as follows. Let X be the open interior of a simple closed curve, and $Me(X)$its medial axis (the set of points reached simultaneously by two or more fire fronts). Let $B(x,\epsilon)$ be an open disk of radius ϵ centered at $x \in X$, and let $R(x)$denote the radius of the largest such disk contained in X. Let $N(x,\epsilon) = Me(X) \cap B(x,\epsilon)\backslash\{x\}$define a "punctured" ϵ-neighborhood of x, one that does not contain x itself. A medial axis point $x \in Me(X)$ is

1. Type 4 if $\exists \epsilon > 0$ *s.t.* $R(x) > R(y)\ \forall y \in N(x,\epsilon)$;
2. Type 3 if $\exists \epsilon > 0$ *s.t.* $R(x) = R(y)\ \forall y \in N(x,\epsilon)$ and $N(x,\epsilon) \neq 0$;
3. Type 2 if $\exists \epsilon > 0$ *s.t.* $R(x) < R(y)\ \forall y \in N(x,\epsilon)$ and $N(x,\epsilon) \neq 0$ and $N(x,\epsilon)$ is not connected; and
4. Type 1 otherwise.

It should be clear that there is a relationship between the above labeling and the velocity function $\frac{dR}{dx}$ along the Medial Axis [50].

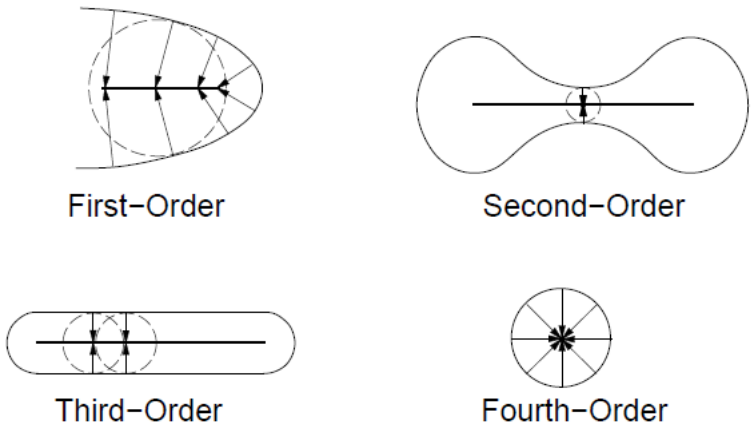

Figure 7: Shock categories. First-order(1-shock): derives from a protrusion, and traces out a curve segment of first-order shocks. Second-order (2-shock): arises at a neck, and is immediately followed by two 1-shocks flowing away from it in opposite directions. Third-order (3-shock): correspond to an annihilation into curve segment due to a bend. Fourth-order (4-shock): an annihilation into a point or a seed. The loci of these shocks gives Blum's Medial Axis.

6.3 The Shock Graph

It is an abstract representation of the shocks in a Medial Axis. The shock types will label each vertex in the graph and the shock formation times (i.e. maximal disk radius or Distance Transform function value) will direct edges to provide an ordering for matching, and a basis for subgraph approximation.

By the shock labeling in the previous section it can be seen that 2-shocks and 4-shocks are isolated points, whereas 1-shocks and 3-shocks are neighbored by other shocks of the same type. To build the shock graph shocks of the same type that form a connected component shall be grouped together, denoting the groups with labels $\tilde{1}, 2, \tilde{3}, 4$,[1] and breaking apart $\tilde{1}$'s at points with three or more generative points (i.e., the points where its maximal disk touches the shape boundary curve). Let each shock group be indexed by a distinct integer i and let t_i denote its time (or times) of formation (i.e., distance transform function value or minimum distance to a shape boundary curve point), corresponding to the radius function evaluated at the shocks in the group. Hence, t_i will be an interval for a $\tilde{1}$; for 2's, $\tilde{3}$'s (i.e., it is a set but all have the same value) and 4's it will be a single number. Finally, let # denote a start symbol and Φ a terminal symbol. The Shock Graph (SG) is a connected graph, rooted at a vertex labeled #, such that all other (non-terminal) vertices are shock groups, and directed edges to non-terminal vertices indicate the genesis of new shock groups.

Formally a Shock Graph of a 2-D shape, $SG(\mathcal{O})$, is a labeled graph $G = (V, E, \gamma)$, with:

- vertices $V = \{1, ..., n\}$;
- edges $(i, j) \in E \subseteq V \times V$ directed from vertex i to vertex j if and only if $i \neq j$, $t_i \geq t_j$, and $i \cup j$ is connected in the plane;
- labels $\gamma : V \rightarrow l$, with $l \in \left\{\tilde{1}, 2, \tilde{3}, 4, \#, \Phi\right\}$; and
- topology such that, $\forall j \in V$ with $\gamma(j) \neq \#, \exists i \in V$ with $(i, j) \in E$

The SG is built in descendant order of the $t_i$values, the distance transform values, because the shocks with higher values correspond to the most significant (central) features. The graph is rooted in the unique vertex labeled #, having as children the last shock groups formed during the grass-fire analogy, i.e., the shock group with the biggest maximal disk. And vertices with label Φ are leaves of the SG, whose parents are the first shock groups to form. Any 2D shape $\mathcal{O}$ has a unique corresponding shock graph $SG(\mathcal{O})$. This uniqueness is proved in [51]. An example of this representation is shown in fig. 6.

6.4 The Shock Graph Grammar

The set of rules presented in the previous section have been grouped according to the semantic processes that they characterize, i.e., the birth, combination and death of shock groups, obtaining a small set of rules shown in figure 8.

[1] $\tilde{}$ stands for a set of points, and the rest cases represent one single point.

The Shock Graph Grammar, SGG, is a quadruple $G(V, \Sigma, R, S)$, with

1. $V = \{\tilde{1}, 2, \tilde{3}, 4, \#, \Phi\}$, the alphabet;
2. $\Sigma = \{\Phi\}$, the set of terminals;
3. $S = \#$, the start symbol; and
4. $R = \{R_1, ..., R_{10}\}$, the set of rules given in figure 8.

The grammar in figure 8 operates by beginning at the start symbol # and repeatedly replacing the left-hand side of a rule by the corresponding right-hand side until no further replacements can be made [52]. The rewrite rules of the Shock Graph Grammar are sufficient to derive the shock graph of any 2D shape $\mathcal{O}$.

Several consequences of these definition:

- Since the same shock cannot be born at two distinct times there exists no path from a vertex back to itself. Consequently, the Shock Graph is a directed acyclic graph (DAG). The problem of searching into acyclic graphs is computationally much simpler than in graphs with cycles on it.
- Since there exist rules in the SGG whose left-hand sides do not consist of single nonterminals, the SGG is not context-free.
- The rewrite rules indicate that a 2-shock and a 4-shock can only be added by rules 5 and 1 respectively, and that semantically equivalent rules exist for a $\tilde{3}$(rules 6 and 1). Hence, a 2-shock and a 4-shock are each semantically equivalent to a $\tilde{3}$ in a specific context. Following this observation, only label types $\tilde{1}$and $\tilde{3}$ have been explicitly assigned. A $\tilde{3}$ with a parent $\tilde{1}$ at each end acts as a 2 (a neck), and a $\tilde{3}$ with a # as a parent acts as a 4 (seed).

Later in [53] a modification of this grammar is proposed in order to include joint points (points where three or more skeleton branches intersect), since they are usually located in the internal part of the skeleton. Consequently they are usually more stable under noise on the boundary curves, so their representation is proved to be more stable under noise than the original definition. Because the points of type 2 and 4 are always isolated points and not point sets, they ignore them in their grammar. Their grammar is $V(1, 3, \cdot, S, T)$, where "1" and "3" correspond to 1-shock and 3-shock groups, "$\cdot$" refers to a joint point, and symbols "S" and "T" correspond to the root and leave nodes respectively. Set $\sum = \{T\}$ contains terminal of the grammar and set R contains rules of the new grammar given in fig. 9. Set $S = \{s\}$ contains start symbol of the grammar.

Each joint point is assigned to a joint node so that all of the end nodes are children of the root node, and the root node has only children of type joint node. Then, for each end point "a" on the skeleton trace skeletal curves branching out from it with identical speed and place shock groups as children of the end node "a". Direction of tracing always is from end points to the terminal points or

other end points. Tracing is stopped at terminal points or when two tracer agents started from two distinct end points, reach to a common shock group. In the last case, common shock group is placed as the child of the two end nodes twice. An example of this kind of shock graph is shown in fig. 10.

6.5 Shock Graph Matching

Given two shock graphs, one representing an object in the scene (V_2) and one representing class template object (V_1), we seek a method for computing their similarity. Unfortunately, due to occlusion and clutter, the shock graph representing the scene object may, in fact, be embedded in a larger shock graph representing the entire scene. Therefore, we have a largest subgraph isomorphism problem, stated as follows: Given two graphs $G = (V_1, E_1)$ and $H = (V_2, E_2)$, find the maximum integer k, such that there exists two subsets of cardinality k, $E_1' \subseteq E_1$ and $E_2' \subseteq E_2$, and the induced subgraphs $G' = (V_1, E_1')$ and $H' = (V_2, E_2')$, are isomorphic. Moreover, since shock graphs are labeled graphs, consistency between node labels must be enforced in the isomorphism.

This graph matching problem is know to be NP-hard for general graphs [54], however, polynomial time algorithms exist for the special case of finite rooted trees [55, 56, 57]. In the next section a method is presented to obtain a unique rooted tree from a shock graph, to improve the efficiency of the matching procedure while keeping consistency. In addition, a depth-first search is performed on the underlying shock trees, to perform a matching beginning with the most meaningful parts of the shape towards its details.

6.6 Shock Graphs to Shock Trees

This section presents a method to convert a DAG representing a shock graph into a unique vertex labeled rooted tree whose size is polynomially bounded by the size of the original shock graph.

Let $G = (V, E)$ be a DAG representing a shock graph on n vertices. A loop L is a subgraph of G formed by the intersection of two directed paths. more formally, L originates at a vertex b, follows two paths P_1 and P_2, and ends at the vertex t. We denote b as the base of L, t as the tip of L, and P_1 and P_2 the wings of L. Due to the shock graph grammar rules the authors of [51] conclude that the tips of all loops are adjacent to nodes having type Φ in G, and each such tip participates in exactly one loop.

The reduction can be obtained therefore as follows. For each tip node t duplicate copies t_1 and t_2 are maintained, and L is redefined to be the union of b and two new disjoint paths $P_1' = P_1 \cup \{t_1\} \cup \{\Phi\}$and $P_2' = P_2 \cup \{t_2\} \cup \{\Phi\}$. This reduction is unique and produces a directed, or equivalently, a rooted tree. This reduction can be computed in linear time, since G has only $O(n)$ tips, and as it is derived from the SGG, consists of checking the in-degree of any $\tilde{3}$'s and 2's, and duplicate them if necessary.

6.7 The Distance Between Two Vertices

Since both shock graphs and trees are labeled, part of the matching procedure includes node label matching, which corresponds to the geometrical properties of shape parts. In this particular case nodes are shock sequences, which are compared one to one. Each shock in labeled by its position, its time of formation (i.e., maximal disk radius), and its direction of flow (or orientation in the case of $\tilde{3}$'s) using the shock detection algorithm in [58].

The main idea is to interpolate a low dimensional curve through their respective shock trajectories, and assign a cost $C(u, v)$ to an affine transformation that aligns one interpolated curve with the other.

Assume that S and S' are two (sampled) shock sequences of the form $S = (s_1, ..., s_p)$ and $S' = \left(s'_1, ..., s'_p\right)$, where each shock point $s_i$is represented by a 4-tuple (x, y, t, α) , corresponding to its Euclidean coordinates (x, y), formation time t, and direction α. For samples from a $\tilde{1}$, the sequence is ordered by time of formation, while for a $\tilde{3}$ there is a partial order to the samples, but no preferred direction. In the latter case, both directions will have to be tried. In order to find the 4D-simplex corresponding to the basis for the affine transformation (in a 4D space) between the two sets, they choose three equidistant points on the chains formed by partial orders $(s_1 \prec ... \prec s_p)$ and $\left(s'_1 \prec ... \prec s'_q\right)$. To preserve the partial order of the points in each sequence, $s_1$should be transferred to s'_1, and s_p to s'_q.

Let (A, B) be the transformation pair for this partial order and, without loss of generality, assume that $p \leq q$. They apply the transformation (A, B) to sequence S to form the sequence $\hat{S} = (\hat{s}_1, ..., \hat{s}_p)$. Once the curves $\Psi\left(\widehat{S}\right)$and $\Psi(S')$, which denote the interpolated 4D curves passing through the points of the sets $\hat{S}$ and S', are defined, the Hausdorff distance measure is computed between them,

$$\Delta\left(\Psi\left(\widehat{S}\right), \Psi(S')\right) = \sum_{x \in \widehat{S}} \inf_{y \in \Psi(S')} \|x - y\|_2 + \sum_{x \in S'} \inf_{y \in \Psi\left(\widehat{S}\right)} \|x - y\|_2 \qquad (10)$$

6.8 Algorithms for Shock Tree Matching

In the original paper describing shape recognition based in Shock Graphs and Medial Axis [51] a depth-first algorithm is used to obtain the maximum subgraph correspondence between two Shock Graphs in their tree form. Depth-first is an algorithm for traversing or searching a tree, tree structure, or graph. One starts at the root (selecting some node as the root in the graph case) and explores as far as possible along each branch before backtracking. See fig. 11 for a graphical representation of the order followed by this procedure. It is therefore an exhaustive exploration paradigm.

Later, Sebastian et al. [59] proposed the use of the edit distance, which is a way to measure the minimum deformations needed to transform each of the shock graphs into the other. This creates a high dimensional search space. They

define a shape cell as a collection of shapes which have identical shock graph topology, and a shape deformation bundle is the set of one-parameter families of deformations passing through an identical sequence of shock transitions. Using this two definitions the search space is partitioned and discretized into cells and the transitions between them are limited to the most simple, the one which includes less cells. In order to obtain this sequence, intermediate shock graphs between both graphs are computed.

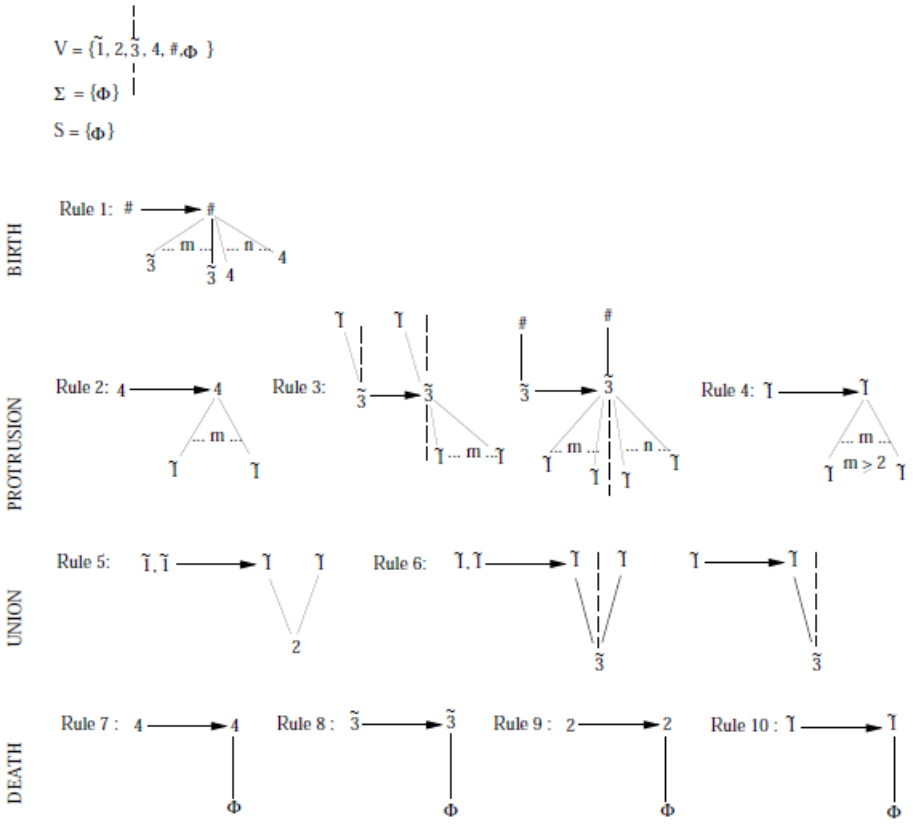

Figure 8: The Shock Graph Grammar, SGG. Dashed lines partition distinct ends of a $\tilde{3}$. The rules are grouped according to the different semantic processes (on the left) that they characterize. Note that the grammar is not context-free, e.g., rule 3 indicates that a $\tilde{1}$ can only be added onto an end of a $\tilde{3}$ that has no parent $\tilde{1}$.

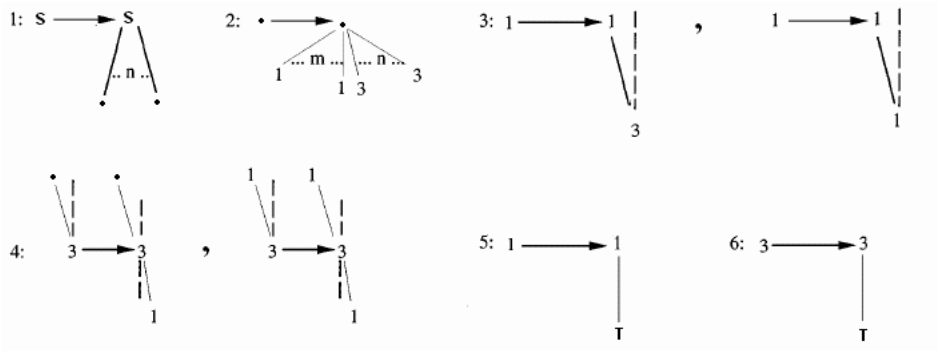

Figure 9: Shock Grammar modification by [53]

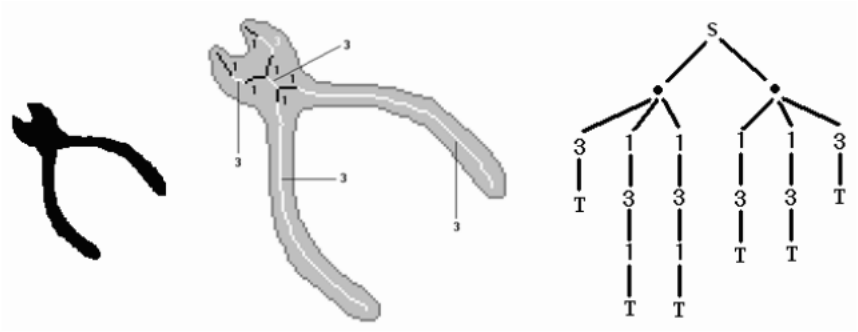

Figure 10: Shock graph example using the modified Shock Grammar in [53]

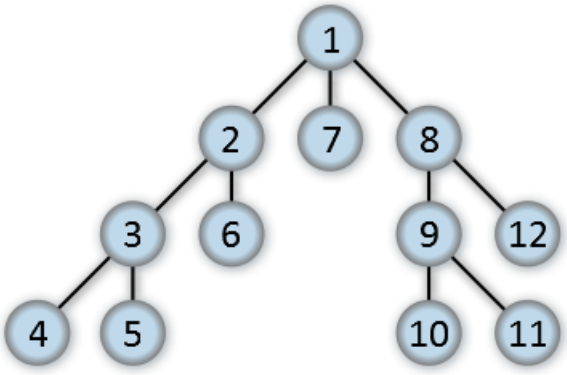

Figure 11: Example of tree traversing order by depth-first procedure. Image courtesy of the Wikipedia

References

[1] G. Borgefors, "Distance transformations in digital images," *Comput. Vision Graph. Image Process.*, vol. 34, no. 3, pp. 344–371, 1986. 2

[2] A. M. Vossepoel, "A note on "distance transformations in digital images"," *Comput. Vision Graph. Image Process.*, vol. 43, no. 1, pp. 88–97, 1988. 2

[3] A. L. D. Beckers and A. W. M. Smeulders, "A comment on "a note on 'distance transformations in digital images"'," *Comput. Vision Graph. Image Process.*, vol. 47, no. 1, pp. 89–91, 1989. 2

[4] I. Sintorn and G. Borgefors, "Weighted distance transforms in rectangular grids," *Image Analysis and Processing, International Conference on*, vol. 0, p. 0322, 2001. 2

[5] F. Leymarie and M. Levine, "Simulating the grassfire transform using an active contour model," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 1, pp. 56–75, 1992. 2

[6] A. Meijster, J. Roerdink, and W. H. Hesselink, "A general algorithm for computing distance transforms in linear time," in *Mathematical Morphology and its Applications to Image and Signal Processing*. Kluwer, 2000, pp. 331–340. 2

[7] Y. Lucet, "A linear euclidean distance transform algorithm based on the linear-time legendre transform," in *CRV '05: Proceedings of the 2nd Canadian conference on Computer and Robot Vision*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 262–267. 2

[8] H. Breu, J. Gil, D. Kirkpatrick, and M. Werman, "Linear time euclidean distance transform algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 529–533, 1995. 2

[9] R. Fabbri, L. D. F. Costa, J. C. Torelli, and O. M. Bruno, "2d euclidean distance transform algorithms: A comparative survey," *ACM Comput. Surv.*, vol. 40, no. 1, pp. 1–44, 2008. 2

[10] I. Ragnemalm, "The euclidean distance transform," Linköping Studies in Science and Technology - Dissertations - No.304, Linköping University, Dept. of Electrical Engineering, Linköping, Sweden, 1993, 276 pages. 2

[11] F. Shih, *Image Processing and Mathematical Morphology*. CRC Press, Inc., 2009, ch. 6. Distance Transformation, pp. 127–181. 2

[12] L. J. Latecki and R. Lakämper, "Convexity rule for shape decomposition based on discrete contour evolution," *Comput. Vis. Image Underst.*, vol. 73, no. 3, pp. 441–454, 1999. 3, 3, 3

[13] ——, "Shape similarity measure based on correspondence of visual parts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 10, pp. 1185–1190, 2000. 3

[14] ——, "Application of planar shape comparison to object retrieval in image databases," *Pattern Recognition*, vol. 35, pp. 15–29, 2002. 3, 3

[15] G. Dirichlet, "Uber die reduction der positeven quadrastischen formen mit drei unbestimmten ganzen zahlen," *Journal of fur die Reine und Angewandte Mathematik*, vol. 40, pp. 209–234, 1850. 4

[16] G. Voronoi, "Nouvelles applications des paramètres continus à la théorie des formes quadratiques premier mémoire: sûr quelques propriétés des formes quadratiques positives parfaits," *Journal für die Reine und Angewandte Mathematik*, vol. 133, pp. 97–178, 1907. 4

[17] ——, "Nouvelles applications des paramètres continus à la théorie des formes quadratiques deuxième mémoire: Recherches sûr les paralléllοèdres primitives," *Journal für die Reine und Angewandte Mathematik*, vol. 134, pp. 198–287, 1908. 4

[18] ——, "Nouvelles applications des paramètres continus à la théorie des formes quadratiques deuxième mémoire: Recherches sûr les paralléllοèdres primitives, second partie: Domaines de formes quadratiques correspondant aux différent types de paralléllοèdres primitives," *Journal für die Reine und Angewandte Mathematik*, vol. 136, pp. 67–181, 1909. 4

[19] A. Thiessen, "Precipitation averages for large areas," *Monthly Weather Review*, vol. 39, pp. 1082–1084, 1911. 4

[20] E. Whitney, "Areal rainfall estimates," *Monrhly Weather Review*, vol. 57, pp. 462–463, 1929. 4

[21] F. Aurenhammer, "Voronoi diagrams—a survey of a fundamental geometric data structure," *ACM Comput. Surv.*, vol. 23, no. 3, pp. 345–405, 1991. 4

[22] B. Aronov, "A lower bound on voronoi diagram complexity," *Inf. Process. Lett.*, vol. 83, no. 4, pp. 183–185, 2002. 4

[23] A. Aggarwal, L. Guibas, J. Saxe, and P. Shor, "A linear time algorithm for computing the voronoi diagram of a convex polygon," in *STOC '87: Proceedings of the nineteenth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1987, pp. 39–45. 4

[24] K. Buchin, "Delaunay triangulations in linear time? (part i)," *CoRR*, vol. abs/0812.0387, 2008. 4.2

[25] E. Bengoetxea, "Inexact graph matching using estimation of distribution algorithms," Ph.D. dissertation, Ecole Nationale Supérieure des Télécommunications, Paris, France, 2003. 5, 5.3

[26] N. Joshi, G. Sita, A. G. Ramakrishnan, and S. Madhvanath, "Comparison of elastic matching algorithms for online tamil handwritten character recognition," in *IWFHR '04: Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 444–449. 5.3.1

[27] H. Bunke and K. Shearer, "A graph distance metric based on the maximal common subgraph," *Pattern Recogn. Lett.*, vol. 19, no. 3-4, pp. 255–259, 1998. 5.3.4

[28] J. Lladoós, E. Martí, and J. J. Villanueva, "Symbol recognition by error-tolerant subgraph matching between region adjacency graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 10, pp. 1137–1143, 2001. 5.3.4

[29] H. Bunke, "Error correcting graph matching: On the influence of the underlying cost function," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 9, pp. 917–922, 1999. 5.3.4

[30] B. Messmer and H. Bunke, "A new algorithm for error-tolerant subgraph isomorphism detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 5, pp. 493–504, 1998. 5.3.4, 5.3.5

[31] K. G. Khoo and P. N. Suganthan, "Evaluation of genetic operators and solution representations for shape recognition by genetic algorithms," *Pattern Recogn. Lett.*, vol. 23, no. 13, pp. 1589–1597, 2002. 5.3.5

[32] S. Auwatanamongkol, "Inexact graph matching using a genetic algorithm for image recognition," *Pattern Recogn. Lett.*, vol. 28, no. 12, pp. 1428–1437, 2007. 5.3.5

[33] R. Myers and E. R. Hancock, "Genetic algorithms for ambiguous labelling problems," in *EMMCVPR '97: Proceedings of the First International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. London, UK: Springer-Verlag, 1997, pp. 345–360. 5.3.5

[34] E. Hancock and J. Kittler, "Edge-labeling using dictionary-based relaxation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 2, pp. 165–181, 1990. 5.3.5

[35] J. Kittler, W. Christmas, and M. Petrou, "Probabilistic relaxation for matching problems in computer vision," in *Computer Vision, 1993. Proceedings., Fourth International Conference on*, May 1993, pp. 666–673. 5.3.5

[36] L. B. Shams, M. J. Brady, and S. Schaal, "Graph matching vs mutual information maximization for object detection," *Neural Netw.*, vol. 14, no. 3, pp. 345–354, 2001. 5.3.5

[37] R. Wilson and E. Hancock, "Structural matching by discrete relaxation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 6, pp. 634–648, Jun 1997. 5.3.5

[38] H.-Y. Kim and J. H. Kim, "Hierarchical random graph representation of handwritten characters and its application to hangul recognition," *Pattern Recognition*, vol. 34, pp. 187–201, 2001. 5.3.5

[39] K. Shearer, H. Bunke, and S. Venkatesh, "Video indexing and similarity retrieval by largest common subgraph detection using decision trees," *Pattern Recognition*, vol. 34, no. 5, pp. 1075 – 1091, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/B6V14-42810JK-C/2/97ba992f6082cb8d8b267ac80fc327ad 5.3.5

[40] D. Rivière, J.-F. Mangin, D. Papadopoulos-Orfanos, J.-M. Martinez, V. Frouin, and J. Régis, "Automatic recognition of cortical sulci using a congregation of neural networks," in *MICCAI '00: Proceedings of the Third International Conference on Medical Image Computing and Computer-Assisted Intervention*. London, UK: Springer-Verlag, 2000, pp. 40–49. 5.3.5

[41] D. Rivière, J. Mangin, D. Papadopoulos-Orfanos, J. Martinez, V. Frouin, and J. Régis, "Automatic recognition of cortical sulci of the human brain using a congregation of neural networks," *Elsevier, Medical Image Analysis*, vol. 6, p. 7792, 2002. 5.3.5

[42] A. Sanfeliu, R. Alquézar, and F. Serratosa, "Clustering of attributed graphs and unsupervised synthesis of function-described graphs," *Pattern Recognition, International Conference on*, vol. 2, p. 6022, 2000. 5.3.5

[43] T. Caelli and S. Kosinov, "An eigenspace projection clustering method for inexact graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 4, pp. 515–519, 2004. 5.3.5

[44] S. Medasani, R. Krishnapuram, and Y. Choi, "Graph matching by relaxation of fuzzy assignments," *Fuzzy Systems, IEEE Transactions on*, vol. 9, no. 1, pp. 173–182, Feb 2001. 5.3.5

[45] Y. Keselman, A. Shokoufandeh, M. Demirci, and S. Dickinson, "Many-to-many graph matching via metric embedding," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 1, June 2003, pp. I–850–I–857 vol.1. 5.3.5

[46] A. Hlaoui and S. Wang, "A new algorithm for inexact graph matching," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 4, 2002, pp. 180–183 vol.4. 5.3.5

[47] M. A. Eshera and K. S. Fu, "An image understanding system using attributed symbolic representation and inexact graph-matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 5, pp. 604–618, 1986. 5.3.5

[48] S. Berretti, A. Del Bimbo, and E. Vicario, "Efficient matching and indexing of graph models in content-based retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 10, pp. 1089–1105, 2001. 5.3.6

[49] P. Lax, *Contributions to Nonlinear Functional Analysis.* New York: Academic Press, 1971, ch. Shock waves and entropy. 6.2

[50] J. Serra, *Image Analysis and Mathematical Morphology.* Orlando, FL, USA: Academic Press, Inc., 1982. 6.2

[51] K. Siddiqi, A. Shokoufandeh, S. Dickenson, and S. Zucker, "Shock graphs and shape matching," Jan 1998, pp. 222–229. 6.3, 6.6, 6.8

[52] H. R. Lewis and C. H.-H. Papadimitriou, *Elements of the Theory of Computation*, 1st ed. Englewood Cliffs, NJ: Prentice-Hall, 1981. 6.4

[53] H. Zaboli and M. Rahmati, "An improved shock graph approach for shape recognition and retrieval," in *AMS '07: Proceedings of the First Asia International Conference on Modelling & Simulation.* Washington, DC, USA: IEEE Computer Society, 2007, pp. 438–443. 6.4, 9, 10

[54] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness.* New York, NY, USA: W. H. Freeman & Co., 1990. 6.5

[55] J. Edmonds and D. W. Matula, "An algorithm for subtree identification," *SIAM Rev.*, vol. 10, pp. 273–274, 1968, abstract. 6.5

[56] R. M. Verma and S. W. Reyner, "An analysis of a good algorithm for the subtree problem, correlated," *SIAM J. Comput.*, vol. 18, no. 5, pp. 906–908, 1989. 6.5

[57] J. Hopcroft and R. Karp, "An n(5/2) algorithm for maximum matching in bipartite graphs," *SIAM J. Comput.*, pp. 225–231, 1975. 6.5

[58] K. Siddiqi and B. Kimia, "A shock grammar for recognition," in *CVPR 96*, 1996, pp. 507–513. 6.7

[59] T. B. Sebastian, P. N. Klein, and B. B. Kimia, "Recognition of shapes by editing their shock graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 550–571, 2004. 6.8