Shape Matching using Skeleton Context for Automated Bow Echo Detection

Mohammad Mahdi Kamani†, Farshid Farhat†, Stephen Wistar‡, and James Z. Wang†
†*The Pennsylvania State University* ‡*Accuweather Inc.*

***Abstract*—Severe weather conditions cause enormous amount of damages around the globe. Bow echo patterns in radar images are associated with a number of these destructive conditions such as damaging winds, hail, thunderstorms, and tornadoes. They are detected manually by meteorologists. In this paper, we propose an automatic framework to detect these patterns with high accuracy by introducing novel skeletonization and shape matching approaches. In this framework, first we extract regions with high probability of occurring bow echo from radar images, and apply our skeletonization method to extract the skeleton of those regions. Next, we prune these skeletons using our innovative pruning scheme with fuzzy logic. Then, using our proposed shape descriptor, *Skeleton Context*, we can extract bow echo features from these skeletons in order to use them in shape matching algorithm and classification step. The output of classification indicates whether these regions are bow echo with over 97% accuracy.**

***Keywords*-Radar image, severe weather forecasting, skeleton pruning, fuzzy logic, big data analytics.**

I. Introduction

Monitoring and storing climatic data around the globe provide a vast amount of data for weather condition analysis. In spite of the fact that computational power is emerging continuously, automatic severe weather forecasting is costly and not always accurate. Meteorologists leverage various and complex models to forecast storms using data from a collection of sensors, including tools and data at the Storm Prediction Center (SPC) of the National Oceanic and Atmospheric Administration (NOAA). The data gathered from these sensors are stored historically; hence it can be leveraged to extract historical patterns of different severe weather conditions. Although meteorologists have developed numerous and complicated models for forecasting storms, they still rely significantly on their interpretations instead of automated algorithms. Further, the majority of these models depend on initial conditions and are highly sensitive to noise, making forecasting much more difficult. Therefore, it is inevitable for this field to combine big data, computer vision, and data mining algorithms with these models to seek faster, more robust, and more accurate results.

Severe weather conditions consist of thunderstorms, tornadoes, floods, lightning, hail, and strong winds. Each of these conditions are investigated widely in meteorological literature, and they need different sources for detection and forecasting, such as satellite images, radar images, temperature, air pressure and wind speed, to name but a few. These events are the primary causes of a large amount of damage around the globe. For instance, according to the *National Severe Storms Laboratory (NSSL)*, damaging winds or *straight-line winds* are the major causes of nearly half of all reports of severe weather conditions in the United States. These winds can reach the speed of 100 miles per hour and have a damage path up to hundreds of miles. Bow echoes, convective line segments with an archery bow shape, are mainly associated with these strong straight-line winds. In some cases, parts of bow echoes can form tornadoes and new thunderstorms. Hence, bow echo detection can be used as a way of forecasting such destructive severe weather conditions. Accurate and on-time forecasting of these events seems necessary and would help to mitigate damages.

As Klimowski *et al.* [1] found 273 cases of bow echoes between 1996 and 2002, it seems popular among weather patterns. Their investigations revealed that bow echoes are causing nearly 33% of severe convectively generated winds in the U.S. [2]. NSSL in partnership with other organizations performed a field experiment on bow echoes called *Bow Echo and Mesoscale Convective Vortex Experiment (BAMEX)* to investigate bow echoes and extremely damaging surface winds with them in more detail [3]. Although, meteorologists have done lots of research on bow echoes and their effects [1]–[4], there is no evidence of computer-aided algorithms in bow echo detection and forecasting in the literature.

In this paper we propose a new method for detecting bow echoes in radar images. This bow shape signature of bow echo leads us to use computer vision algorithms for particular shape detection and matching. In this regard, we first use radar images and develop our algorithm for skeletonization, which then can be used in the shape matching algorithm through our suggested descriptor, *Skeleton Context*. Finally, we use *Mixture Discriminant Analysis (MDA)* to classify bow echo shapes in radar images. Our **contributions** through this research are as follows:

- Introducing a new skeletonization scheme and some criteria for ranking of edges in a skeleton.
- Proposing a novel fuzzy logic based approach for *skeleton pruning*, which is based on inference systems that are closely related to the human inference system. The process is flexible due to the suggested *branch tolerance* window. We introduce two fuzzy transformations of skeletons in the course of pruning them, that is, *Main Skeleton Degree of Belief*, and *Branch Degree of Belief*.

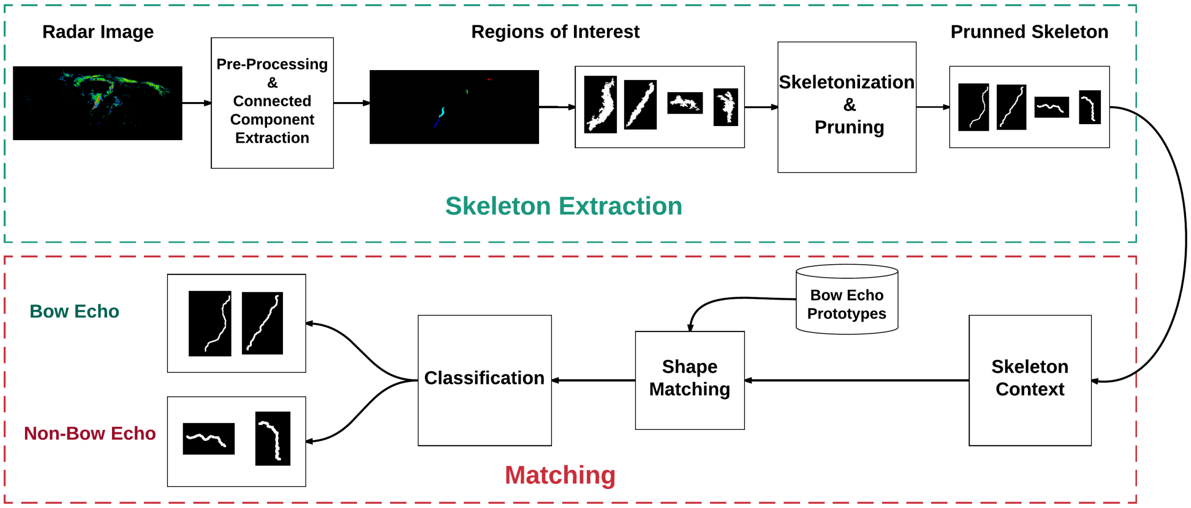


Figure 1: The complete scheme for automatic bow echo detection. In the first stage, it extracts regions of interest, which are red parts in radar images. Then in the skeletonization step, it extracts skeleton map of each part, and prunes it using the proposed algorithm. After finding pruned skeleton, it computes skeleton context and finds the nearest match in the database of bow echo prototypes. Finally, it uses mixture discriminant analysis classifier to detect whether it is a bow echo or not.

- Introducing a new shape descriptor, based on skeleton samples of a shape, called *skeleton context*. Applying a novel approach on shape matching algorithm, in order to allow *partial shape matching*.

In Section II, we describe bow echoes and the dataset. Then we propose our algorithm for skeletonization and skeleton pruning, and define Skeleton Context. Section III is dedicated to classification part and shape matching scheme. Section IV provides evaluation results for classifying bow echoes, and finally we conclude in Section V.

II. Bow Echo Feature Extraction

A. Overview

Severe weather conditions such as thunderstorms, tornadoes, hail, and especially strong straight-line winds are associated with bow echoes. The wind with a bow echo can be fierce and reach violent intensity.

The term bow echo was coined by Fujita [5], to describe strong outflow winds associated with storms that spread out in straight lines over the ground. Przybylinski categorize bow echoes in two categories [4]:

- Bow echo patterns associated with derechoes or straight-line winds.
- Bow echo patterns associated with vortices, including tornadoes,

Klimowski *et al.* [1] classify different types of bow echoes and their evolution from meteorologists' point of view. They start with a radar echo and then evolve into a bow echo. In this research, we aim to use this topological feature of these phenomena to detect them using computational approaches.

Our proposed scheme for detecting bow echoes, shown in Figure 1, consists of two main steps, skeleton extraction and matching. In the first step, we take a radar image and extract its regions of interest (parts that we can find bow echoes). Then using our skeletonization and skeleton pruning framework, to extract their skeleton. In the second step, using our suggested shape descriptor, skeleton context, to extract features for shape matching part. After matching those parts to a bow echo prototype, based on the distance between them and their matched bow echo prototype, we are able to identify whether they are a bow echo or not.

B. Radar Images and Regions of Interest

Our dataset consists of images from NEXRAD level III radar of National Weather Service (technical name WSR-88D), which can measure precipitation and wind movement in the atmosphere. These images are gathered from 160 active high resolution radar sites around the U.S. continent. We use base reflectivity images from NEXRAD level III radar, which represent the amount of returned power to the radar from transmitted signal after hitting precipitation in the atmosphere. The images have 4-bit color map with $6,000 \times 2,600$ pixels of spatial resolution, which are stored every five minutes [6]. The color map associated to these radar images is shown in Figure 2 having the range from 0 dBZ to 75 dBZ for reflectivity. The range of the reflectivity from 0 dBZ to -30 dBZ, alongside with "No Data" regions (due to spots with beam blockage in the mountains and outside of the U.S.) is represented by a black color.

Bow echoes can be spotted in heavy precipitation red regions on radar images (*i.e.*, with reflectivity of higher than 50 dBZ). As it is shown in Figure 2, the bow echo happened on May 2 in 2008 over Kansas City. Hence, in searching for bow echoes in radar images, regions of interest are red

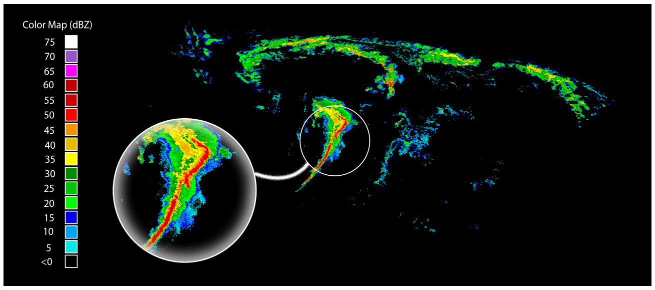


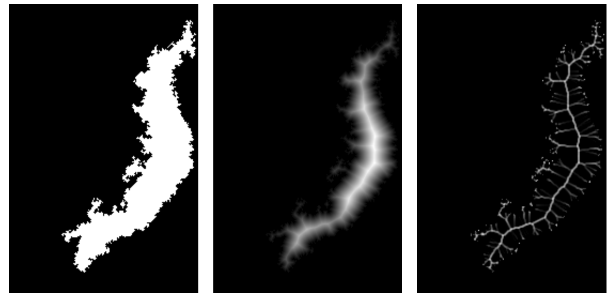
Figure 2: Radar image of the United States Continent with a bow echo, May 2, 2008, 07:10 GMT. We magnify the part that bow echo happened (*i.e.* red regions).

in color ($>$ 50 dBZ). To extract these parts we can set a threshold on their RGB values, but it would result in patchy areas and not connected regions. Although human eyes can cluster them as a unified region, computer algorithms need to perform a pre-processing in order to connect those parts together. We use some morphological operations such as image closing along with active contours to improve the extraction of the connected components.

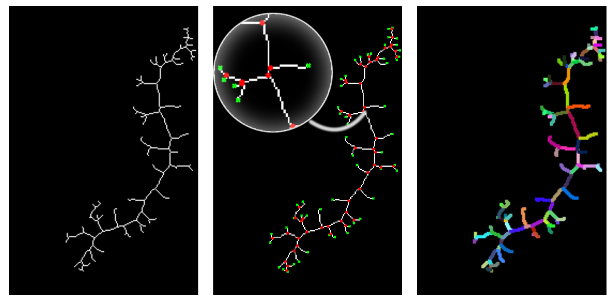
C. *Skeletonization*

Skeleton of a shape is a low-level representation that can be used for matching and recognition purposes in various fields of study including image retrieval and shape matching [7]. Skeleton can provide a good abstraction of a shape, which contains topological structure and its features. Since it is the simplest representation of a shape, there has been an extensive effort among researchers to develop generic algorithms for skeletonization of shapes [8]–[12]. Specifically the vast majority of the algorithms are based on Blum's "Grassfire" analogy and formulation for skeletonization [13] . The most important key factor in skeletonization algorithms is to preserve the topology of the shape. One of the most widely used algorithms is based on measuring the net outward flux by using Euclidean Distance Transform (EDT) of the binary image followed by a topology preserving thinning algorithm [11]. We use the method introduced by Dimitrov *et al.* [11] to calculate the net outward flux per unit area and detect the location of the pixels where conservation of energy principle is violated. EDT maps a binary image into a gray level image with value of each pixel represents its euclidean distance to the border of image. Given Euclidean distance of an image (D_E), we should first compute the gradient vector field (∇D_E), and then the divergence of this vector field [11]. Mathematically, the divergence of the gradient vector field ($\nabla \cdot (\nabla D_E)$) is defined as the limit of the net outward flow of the field across the boundary of the area around the given point, while the area is shrinking to zero:

$$\nabla \cdot (\nabla D_E) = \lim_{S \to 0} \iint_C \frac{\nabla D_E \cdot \vec{n}}{S} dC \ , \tag{1}$$



(a) Binary image (b) EDT (c) Flux map



(d) Initial skeleton (e) Graph vertices (f) Graph edges

Figure 3: Different stages in the skeletonization process.

where C is the boundary, S is the area, and $\vec{n}$ is the normal vector of the boundary. Hence, we can calculate net outward flux at each point $P = (x, y)$ as follow:

$$Flux(P) = \sum_{i=1}^{8} \nabla D_E(Q_i) \cdot \vec{n} \ , \tag{2}$$

where Q_i's are neighbor points to point P. According to the direction of the normal vector, we can determine that positive or negative flux values are representing drain or source of energy, where energy-draining points are internal skeletal points, and energy-generating points are external skeletal points. In the Figure 3b, the EDT of the binary image, in Figure 3a, is shown. Then, in Figure 3c the net outward flux for this Euclidean distance transform is computed.

Setting a threshold on flux values, initial binary skeleton map can be computed as depicted in Figure 3d. Since we are dealing with highly boundary-variant shapes, the skeleton map would contain a large number of unwanted branches that make the subsequent matching steps complicated. We need to develop an automated method to remove all such branches while keeping the main skeleton intact. We introduce a new method to prune the skeleton using fuzzy logic, which will be discussed in the next section. Our pruning algorithm needs to have the complete graph information of the skeleton including its vertices and edges' pixels coordinates. Therefore, the skeleton map is converted to a graph before the pruning step. To convert a skeleton map to a graph, we start finding vertices in the map. Hence, for each skeleton point on the map we find 8-connected neighbors.Then in each 3×3 matrix of neighbors, we construct the graph of pixels , in which pixels with binary value of 1 are vertices. In this graph, edges represent the connection of vertices with their neighbors in 4-connected positions. Next, we find the Euler characteristic of this graph, which is number of vertices minus number of edges in a 2D graph. If Euler characteristic is greater than two, the point is a branch point, and if it is equal to one the point is an end point. Vertices extracted in this way for the skeleton in Figure 3d, is depicted (with a magnified part for a better representation) in Figure 3e. Red points in the image are branch points, and green ones are end points.

After finding vertices in the graph, we should form the edge list of the graph. We start with a random end point and traverse its neighbors to reach a branch point or other endpoints. If the neighbor pixel is not a branch point nor an endpoint, it would be added to the current edge's pixel list. When we reach a branch point, we should add another edge to the edge list, having that branch point as its first point, and start searching edge points for the next edge in the edge list. If we reach an end point, we just start searching edge points for the next edge in the edge list. This approach will be continued until there is not any edges left unprocessed. Detailed algorithm of transforming skeleton map to graph is in Algorithm 1, and the result of finding edge list is shown in Figure 3f with different colors for different edges.

D. Skeleton Pruning using Fuzzy Logic

Having high sensitivity to border variations, almost all algorithms for skeletonization need to be followed by a pruning stage in order to remove thin branches caused by boundary deformations. These branches may significantly change the skeleton graph, and hence they should be treated carefully for the matter of topology preserving in skeletonization algorithms. This issue would be intensified in case of radar images and bow echo shapes, because they have drastic variations on their borders, as it is evident in radar images. There has been studies [14]–[16] investigating direct as well as indirect methods to address this issue. Most of these algorithms use Boolean logic in their decision to remove or keep the branches. The output of these algorithms is a crisp value attributed to each edge distinguishing branch edges from main skeleton edges. However, if we ask a person to do pruning on a skeleton graph, he or she would extract the main skeleton with an uncertainty to some extent. On the other hand, fuzzy logic introduces many-valued logic in close proximity to human decision making system [17], [18]. Hence, we propose an approach based on fuzzy inference system to prune skeleton graph and extract the main skeleton.

In our method, we use the outward flux values of the pixels as an input to the fuzzy inference system. Heuristically from raw images of flux values (Figure 3c), the higher the value of the outward flux in each pixel, the more probable that the pixel is in the main skeleton. Therefore, based on

Algorithm 1 Skeleton2Graph

Input: Skeleton Map
Output: Branch Points, End Points, Edges List

```
procedure FIND VERTICES
    for all Points on Skeleton do
        PointMatrix ← find 8-Connected Neighbors
        Filter PointMatrix by a 4-connected Neighbors mask
        EulerCharacteristic ← #Vertices − #Edges
        if EulerCharacteristic > 2 then
            BranchPoints ← Point
        else if EulerCharacteristic = 1 then
            EndPoints ← Point
        end if
    end for
end procedure
procedure CREATE EDGE LIST
    EdgeList{1} ← Select one End Point randomly
    EdgeNumber ← 1
    while EdgeNumber ≤ # Edges in EdgeList do
        SearchPoint ← EdgeList{EdgeNumber}(end)
        SearchMatrix ← 8-Neighbors Connected to SearchPoint
        for all points in SearchMatrix do
            Set Value to −EdgeNumber
            if A point is in BranchPoints then
                Add a new edge to EdgeList with having this Branch point as its first Point
                EdgeNumber ← EdgeNumber + 1
            else if A Point in EndPoints then
                EdgeNumber ← EdgeNumber + 1
            else
                Add Points to EdgeList{EdgeNumber}
            end if
        end for
    end while
end procedure
```

this observation we can extract a feature for every edge connected to a vertex in the skeleton graph. In order to record vertices properties in the flux images, we form an array, called Γ, for each vertex with the length of the number of the edges linked to it. The value attributed to each edge e_j connected to the vertex V_i could be computed as follow:

$$\Gamma\{V_i, e_j\} = \frac{1}{M_j - 1} \sum_{P_j=2}^{M_j} \left(W_G(P_j) \cdot Flux(P_j)\right), \qquad (3)$$

in which, $j = 1, ..., N_i$ indicating the j^{th} edge connected to V_i. N_i is the number of edges linked to V_i, M_j is the number of pixels in the j^{th} edge, and $W_G(P_j)$ is the Gaussian weight for each pixel of edge e_j computed as follow:

$$W_G(P_j) = \exp\left(-\frac{||P_j - V_i||^2}{2\sigma^2}\right). \qquad (4)$$

Our proposed fuzzy inference system (FIS) consists of two components:

- FIS-1: Fuzzy inference system to compute degree of belief of each pixel to main skeleton edges.
- FIS-2:Fuzzy inference system to compute degree of belief of each pixel to branch edges.

The FIS-1 output, indicates that to what extent we believe an edge belongs to main skeleton. Afterwards, we use this value as an input to FIS-2 to compute the extent to which that we believe an edge belongs to branch edges. These values are the same for the pixels of the edge and varies among different edges. In following subsections we introduce these two fuzzy inference systems, their inputs, rules, and outputs. And then we go through the details of our algorithm for pruning skeleton.

Main Skeleton Fuzzy Inference System

We now describe the inputs, outputs, and the operators used in the first fuzzy inference system (FIS-1).

Fuzzy Inputs: FIS-1 has two inputs as follows:

Importance Value (I) : which indicates the importance of each edges, and is generated from vertices properties linked edges. The details of computing importance value would be described later. Based on variations in the importance values of different vertices, we can calculate the expected value and variance of this feature in an image. Thus, the universe of the discourse for this input feature could be in the range of $[E\{I\} - 6 \times \sigma_I, E\{I\} + 6 \times \sigma_I]$, where $E\{.\}$ is the expected value operation, and σ_I is its standard deviation. In this universe of discourse we define three different fuzzy sets including, *Low*, *Medium*, and *High*, each of which have Gaussian membership functions with standard deviation (sigma) equal to σ_I and different central values.

Edge Length (L_E) : One of the most important features for detecting main skeleton edges, is edge length. However, branches mostly happen where boundary is deformed with variations, and hence the skeleton would be furcated into too many small branches. As a result, we can use the edge length as an indicator for main skeleton pixels, alongside with other factors. Edge length in each image is random variable that we can find its expected value as $E\{L_E\}$, and its standard deviation as σ_{L_E}. Hence, the universe of the discourse for edge length input feature would lie in the region of $[E\{L_E\} - 5 \times \sigma_{L_E}, E\{L_E\} + 5 \times \sigma_{L_E}]$. Just in this case, we should make sure that the minimum value for edge length is not less than zero. For this input, we consider three fuzzy sets with Gaussian membership functions, all with the same sigma value, σ_{L_E}, and different centers. These fuzzy sets are named *Small*, *Medium*, and *Long*.

Fuzzy Output: This FIS has one output, that is, *Main Skeleton Degree of Belief* (Ψ_{MS}). This output represents the degree of belief on pixels to be on the main skeleton graph. The range of its value is between $[0, 1]$, and we define 5 different Gaussian membership functions as its fuzzy sets. The sigma value for these fuzzy membership functions is set to 0.05. These functions are as follow: *Very Low*, *Low*, *Average*, *High*, and *Very High*. The output resulted from this FIS for the skeleton in Figure 3d is shown in Figure 4a. The higher the value in image, the higher degree of belief of main skeleton (Ψ_{MS}) on that edge.

Fuzzy Operators: In each fuzzy inference system we should define methods of integration of membership functions. First of all we should decide about the method of integrating different inputs in each rule, then the method for implication of the output in each rule, and at the end the method of aggregation of outputs from each rule. Hence, we choose these operators as follow:

- Fuzzy Operation: we choose the simplest method, that is, ***min*** for ***AND*** operations and ***max*** for ***OR*** operations.
- Implication Method: for the implication of the output we choose ***min*** operator.
- Aggregation: for the aggregation of the outputs, we use ***max*** operator.

Branch Fuzzy Inference System

The details of the second fuzzy inference system (FIS-2) are provided below.

Fuzzy Inputs: FIS-2 has three inputs including the output of FIS-1.

Main Skeleton Degree of Belief (Ψ_{MS}) : This is the output from the first FIS. However, for this FIS, we only consider two fuzzy membership functions, namely, *Low* and *High*. Instead of Gaussian, we choose trapezoid membership functions this time.

Edge Length (L_E): Because of the importance of edge length in distinguishing between main skeleton and branch edges,we use this feature in our second FIS, with the same range for its universe of discourse. But we merely define two membership functions in FIS-2 for this input, consisting of two trapezoid functions named *Small* and *Long*.

Curvature Score (S_C) : In the course of searching for the main skeleton, we may encounter with a vertex that has two output edges, in which almost all of their properties are similar. The only difference between these two edges is their angle with the nearest main skeleton edge. The more this angle is close to zero, the more probable that we categorize the edge as a main skeleton edge rather than a branch edge. Therefore, we introduce Curvature Score for vertex V_i as follows:

$$S_C^{V_i}(\vec{e_{i_1}}, \vec{e_{i_2}}) = \cos(\theta_{i_1,i_2}) = \frac{< \vec{e_{i_1}}, \vec{e_{i_2}} >}{||\vec{e_{i_1}}|| \; ||\vec{e_{i_2}}||}, \qquad (5)$$

where $\vec{e_{i_1}}$ is the vector starting from the middle point of reference edge (or main skeleton edge) to V_i, and $\vec{e_{i_2}}$ is the vector starting from V_i to the middle point of the test edge. The universe of discourse for this input would lie in the range of $[-1, 1]$, and we choose two trapezoid membership functions on this range with the name of *Averted* and *Straight*.

Fuzzy Output: The fuzzy output of this FIS is representing the degree of belief on edges to be member of branch edges, which is in the range of $[0, 1]$, and we call it Branch Degree of Belief (Ψ_B). We consider three membership functions for this output, namely, *Low*, *Average*, *High*, in which the first and the last one are trapezoid and the second one is a triangle membership functions. The result of this FIS on the skeleton of Figure 3d is depicted in Figure 4b. Higher values in the image shows higher branch degree of belief (Ψ_B).

Fuzzy Operators: We use the same fuzzy operators as the first FIS for this FIS, that is, ***min*** for ***AND*** operation and ***max*** for ***OR*** operations in the rules, ***min*** operator for output implications, and ***max*** operator for aggregations.

Importance Value (I)

As it was mentioned in FIS-1, we have to compute value I for each edge based on their flux values. Thus, we start with the best edge in the sense of the highest flux value (the average of Γ values of its both vertices), and then continue to move from both ends toward other edges, until all the edges are covered. Since we are computing the importance value of each vertex in a predefined direction, it could be considered as a "Tree" with vertices and edges in a hierarchical manner. Hence, in the course of computing importance value of a vertex we could include flux values of edges in the lower level linked to that vertex in the hierarchy. Sometimes flux values of edges on the main skeleton decrease (in proportion to their linked edges), that pruning algorithm does a false detection, if we merely rely on flux values of each edge. Thus, this operation ensures to choose the main skeleton edges by increasing their importance values. As a result, we want to add depleted version of importance value of lower level vertices to importance value of current vertex. Hence, we can calculate the importance value for the vertex V_i in the recursive scheme as follows:

$$I(V_i, \Omega) = \begin{cases} \left(\gamma e^{-0.3}\right) \sum_{k_i=1}^{M_{i,\Omega}} I(V_{k_i}, \Omega + 1), & \Omega < \Omega^\star \\ \left(\gamma e^{-0.3}\right) \sum_{k_i=1}^{M_{i,\Omega}} \sum_{j_i=1}^{N_{i,\Omega}} \Gamma\{V_{k_i}, e_{j_i}\}, & \Omega = \Omega^\star \end{cases} \tag{6}$$

in which, Ω shows the depth level (with $\Omega = 1$ the closest level to vertex V_i), and $\Omega^\star$ is the maximum depth level defined by user. $\left(\gamma e^{-0.3}\right)$ is damping factor, $M_{i,\Omega}$ is the number of vertices linked to V_i in the level Ω, and $N_{i,\Omega}$ is the number of edges linked to V_i in the level Ω (when $\Omega \geq 2$, the connection is indirect). In summary, for calculating the importance value of a vertex we should consider vertices properties (Γ) of all subsequent vertices below this vertex in the tree.

Pruning Algorithm

Using fuzzy inference systems with proper importance values of vertices and edges, the functionality of the pruning algorithm would be straight forward. We just need to define some parameters to give users ability to decide to what extent they tolerate branches in an image and the maximum threshold of Ψ_B on an edge, to consider it as a branch. We define these parameters as follows:

$$\begin{aligned} B_T &\equiv \text{Branch Tolerance} \in [0, 1]\,, \\ \Psi_B^\star &\equiv \Psi_B \text{ Threshold} \in [0, 1]\,, \end{aligned} \tag{7}$$

where $B_T = 0$ means we do not tolerate any branches and we just want the medial axis. For example, if $B_T = 0$ and $\Psi_B^\star = 0.6$, then we would like to get rid of all edges that have Ψ_B greater than or equal with 0.6. If B_T is set to 1, it does not mean that we want to keep all edges with Ψ_B less than $\Psi_B^\star$, but we want to constraint the decision based on the situation of each branch point. Take for instance, a branch point has 2 edges with both Ψ_B values less than $\Psi_B^\star$ and $B_T = 1$, but there is a huge difference between their values, which makes us to choose the one with the lesser value. Hence, based on these two parameters and Ψ_B values in each branch point separately, we should form a Belief Window (BW) to filter desired values on that particular branch point. This window should always start from the minimum value among Ψ_B values of edges linked to a branch point, and can have a size in the form of equation below for the vertex V_i and set of edges linked to it in the lower level E_i:

$$\begin{aligned} &BW(V_i, E_i) = \\ &B_T^{\Delta(V_i,E_i)} \times \Delta(V_i, E_i)^{f_1(B_T)} \times e^{f_2(\Delta(V_i,E_i))}\,, \end{aligned} \tag{8}$$

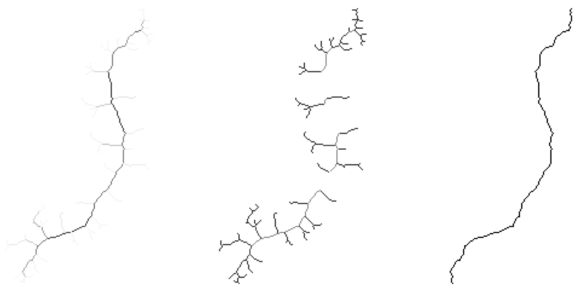
where

$$\begin{aligned} \Delta(V_i, E_i) &\triangleq \Psi_B^\star - \min_{E_i} \Psi_B\,, \\ f_1(B_T) &\triangleq \alpha + \beta B_T\,, \\ f_2(\Delta(V_i, E_i)) &\triangleq \kappa + \lambda \Delta(V_i, E_i)\,. \end{aligned} \tag{9}$$

This function would satisfy aforementioned property of belief window for choosing branches. Parameters of lines in f_1 and f_2 can be set heuristically. For example, we can set them as follow: $\alpha = 1.5$, $\beta = 0.2$, $\kappa = 1.5$, and $\lambda = -2.5$. When this window is formed, in each branch point we can decide which edges to keep or omit. The pruned skeleton of Figure 3d is in the Figure 4c.

E. Skeleton Context

Shape Context [19] as a powerful shape descriptor represents a rough distribution of all other points with respect to a selected point in terms of distance and angle. Shape context is used to find correspondences between samples from border of two shapes, and then find the cost of



(a) (b) (c)

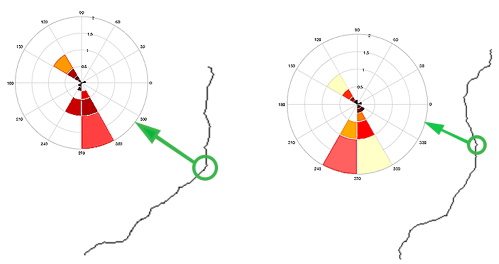
Figure 4: Output of pruning algorithm: (a) main skeleton degree of belief, (b) branch degree of belief, (c) Pruned skeleton. In (a) and (b) images are in grayscale, and higher values represent higher degree of beliefs

matching two shapes using bipartite graph matching. After that, parameters for an affine transform are extracted using thin plate spline (TPS), in order to map points in one shape to their correspondences in the other shape with warping the coordinates. Finally, a notion of shape distance for recognition purposes is exploited.

As there are a lot of fluctuations over the boundaries of the shapes in radar images, these make object matching with boundary samples less effective, and it may result in false matching. On the other hand, matching objects using skeleton samples sounds more robust in the sense that pruned skeleton contains complete shape topology regardless of its boundary variations. Hence, we use *shape context* to introduce a new descriptor called skeleton context. As it is shown in an example in Figure 5, skeleton context is log-polar histogram, formed for each sample point on the skeleton. For each sample point P_i, the center of this log-polar histogram is located on that sample point, then each bin in the histogram represents the number of sample points in the specific angle and range of distance from the center (i.e. P_i) determined by that bin. We use the notation of $H_{SC}(P_i, r_{k_1}, \theta_{k_2})$, to show the value of skeleton context's histogram for point P_i, in the (r_{k_1}, θ_{k_2}) bin. These histograms contain information of each sample point in proportion to other sample points on the skeleton of an object, and hence it could play object descriptor role for shape matching. As a result we could use these descriptors as feature data for bow echo detection in the next step. In Figure 5 you can find the skeleton context computed for two points of two objects, which matched together in our algorithm.

III. Bow Echo Classification and Detection

With skeleton context defined in previous section, we are able to extract features from objects in radar images and use them in the recognition process. In the following subsections, we introduce our proposed model for learning



(a) (b)

Figure 5: Skeleton Context of 2 points on different skeletons that are matched based on the algorithm.

features of bow echoes and implementing a classifier in order to detect bow echoes in tons of objects extracted from radar images.

A. Shape Matching with Skeleton Context

The procedure for shape matching is nearly the same with the method introduced in [19], that is, instead of shape context, we use our proposed descriptor skeleton context. Defined in [19], we can compute the cost of mapping each skeleton point P_i^1 in image 1, to each skeleton point P_j^2 in image 2 as follows:

$$C(P_i^1, P_j^2) = \frac{1}{2}\sum_{k_1,k_2} \frac{\left(H_{SC}(P_i^1, r_{k_1}, \theta_{k_2}) - H_{SC}(P_j^2, r_{k_1}, \theta_{k_2})\right)^2}{H_{SC}(P_i^1, r_{k_1}, \theta_{k_2}) + H_{SC}(P_j^2, r_{k_1}, \theta_{k_2})}\ . \tag{10}$$

Having the cost of all possible mapping, we can use one of the algorithms designed to solve the bipartite graph matching, such as Hungarian method [20], which finds the minimum cost solution for matching points in image 1 to image 2. We define a threshold for the cost of matching, which indicates that if the minimum cost of matching one point from image 1 to the points of image 2 is greater than that threshold, then we announce that there is no matching point for this sample point in image 1. This could be applied, by adding some dummy points with matching cost of defined threshold, which allows some points (at most equal to the number of dummy points added) to have no matches in the other shape. This is necessary for having partial shape matching, which would be described in the next section. After that, our affine transformation can map in the last step by warping coordinates. As mentioned, for this step we could use TPS interpolation which tries to minimize the bending energy as it is defined in [19]. Warping coordinates, we need an indicator that shows the intensity of changes on shapes in the transformation . For instance, if the transformation is just a simple rotation or relocation, this indicator should be low, which means the transformation has not warped coordinates extensively. But it would be high, if the transformation is warping coordinate significantly to map the points together. If the affine transformation could be written in mathematical form as:

$$\vec{y} = f(\vec{x}) = A\vec{x} + \vec{b}\,, \tag{11}$$

with matrix A as linear map and vector $\vec{b}$ as the offset for translation, then we can write our indicator for affine transformation as follows:

$$\mathcal{C}_A = \log \frac{\sigma_1(A)}{\sigma_2(A)}\,, \tag{12}$$

where $\sigma_i(A)$ shows i-th singular value of matrix A, with $\sigma_1 \geq \sigma_2$. The more $\mathcal{C}_A$ is closer to zero, the more the skeleton of two shapes are similar, and the less coordinates are warped. In total, we can use four different indicators for the quality of matching of two shapes using skeleton context.

- *Matching Cost ($\mathcal{C}_{MC}$)* : This is defined based on the cost of matching points in two shapes, as it was computed in equation 10.
- *Bending Energy ($\mathcal{C}_{BE}$)* : The energy that TPS wants to minimize, which is described in [19].
- *Affine Indicator ($\mathcal{C}_A$)* : which is defined in equation 12.
- *Matching Ratio ($\mathcal{C}_{MR}$)* : It represents the ratio of the number of sample points in image 1, that we could match with points in image 2; to the total number of sample points.

B. Neighboring Effect

As it was discussed, we need partial shape matching, as bow echoes have a tail in addition to the bow part in some cases, and this tail might be different among various bow echoes. Therefore, we should add an alteration to the shape matching part, in order to include partial shape matching in our algorithm. To do so, after the first iteration of shape matching, we can learn the initial mapping between sample points of two images. Since in our algorithm for extracting edge list, points are listed in accordance with their spatial order, we can use this ordinal positions of sample points to define neighbors. For instance, if $(m-1)$-th and $(m+1)$-th sample points in image 1 are mapped respectively to $(n-1)$-th and $(n+1)$-th sample points in image 2 , then we expect m-th sample point in image 1 to be mapped as close as possible to n-th sample point in image 2. This is what we call neighboring effect, and we want to impose this constraint in the shape matching algorithm by adding a Neighbor Cost to the cost introduced in equation 10. This Neighbor Cost could be in the form of:

$$\mathcal{C}_N(m) = \epsilon_N \left[1 - \exp\left(-\frac{(m - E\{\delta_N\})^2}{2\sigma^2} \right) \right], \tag{13}$$

where ϵ_N is the maximum amplitude of this cost, and random variable δ_N is the difference between sample points' numbers in image 1 with respect to the their mapped sample points' in image 2. This Gaussian shape function would try to keep mapping of each sample point in accordance to its neighbors.

C. Classifier

After defining features for detecting a bow echo in section III-A, we should learn a classifier for the classification part.First we should select some prototypes from our bow echo samples, and then we can use them in the next stages of classification as representatives of bow echo class. Having abundant types of bow echoes in radar images, we can use *K-medoids* algorithm [21], in order to find prototypes in bow echo class. Thereafter, these prototypes are used as a reference for the bow echo class. Hence, to decide whether a skeleton image belongs to bow echo class or non-bow echo class, we just need to find the minimum distance between skeleton image and bow echo prototypes. The distance of two skeleton is defined as a linear combination of four different costs as they were characterized in section III-A. As a classifier we use Mixture Discriminant Analysis (MDA) [22], which is generalized version of Linear Discriminant Analysis (LDA). In MDA, we consider each class has R_m prototypes with Gaussian distributions, and next, using Expectation Maximization (EM) algorithm [23] to find the Gaussian-distributed parameters and probability of each sample in each Gaussian-distributed sub-class. Thus, the EM algorithm alternates between these two steps [22]:

- *Expectation Step* (E-step): Given the parameters for the distributions of R_m sub-classes in class m, we should assign a weight for each sample in each sub-class, with total sum of probabilities equal to one.
- *Maximization Step* (M-step): Using weights computed in the E-step, in order to compute weighted Maximum Likelihood estimation for parameters of each sub-class distribution.

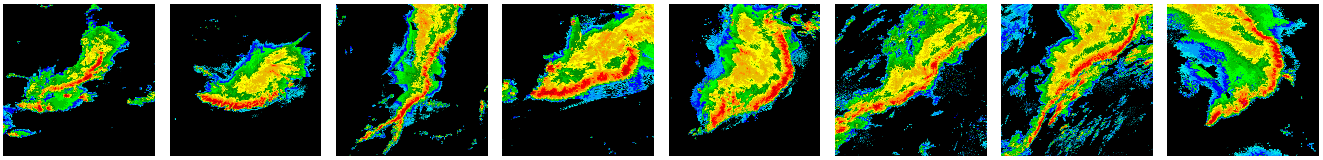
After training and finding the parameters for sub-class distributions in both bow echo and non-bow echo classes, our classifier assess the weight of new coming samples for each subclass in two classes. This would determine the class of that sample based on the label of the nearest subclass to the sample.

IV. Case Study

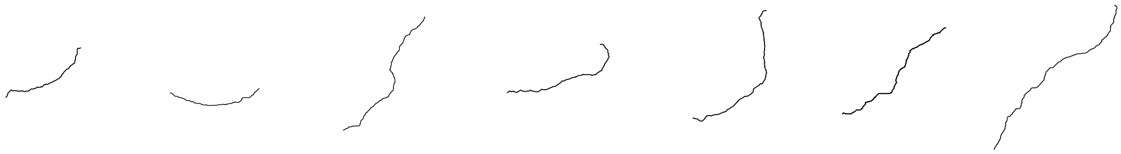
Using skeletonization approach and shape matching algorithm with skeleton context introduced in section II and section III respectively, alongside with the classifier defined above, we are able to detect bow echoes in radar images automatically with high rate of accuracy. To test this approach and classifier, we build a database for a case study.

A. Training Data

Our database consists of US radar images in 2008. We have chosen that year because it had a large number of severe weather activities. The techniques developed, however, are general and applicable to any year. We search the whole year



(a) 2008/08/18 (b) 2008/06/26 (c) 2008/05/02 (d) 2008/08/04 (e) 2008/05/24 (f) 2008/01/11 (g) 2008/02/06 (h) 2008/05/23



(i) 2008/08/18 (j) 2008/06/26 (k) 2008/05/02 (l) 2008/08/04 (m) 2008/05/24 (n) 2008/01/11 (o) 2008/02/06 (p) 2008/05/23

Figure 6: Bow echo prototypes with $k = 8$. The top row is radar images, and below their respective skeleton extracted.

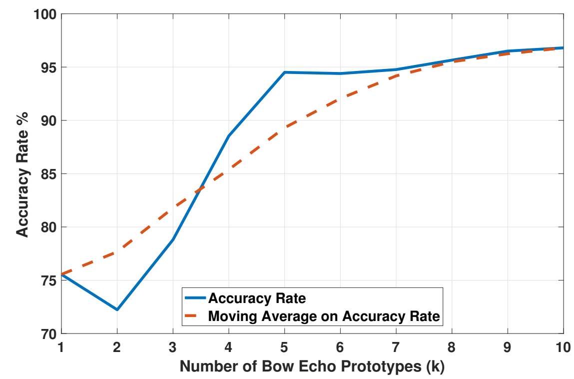
images to find those dates that bow echo happened in United States, and succeed to label 89 distinct days with bow echo during 2008 including $1,148$ radar images. After that, we extract skeleton of regions of interest in each radar images and label them as bow echo and non-bow echo classes. In each image that we can spot a bow echo, there would be some other parts that are not bow echo but would be captured as regions of interest, for their reflective signal amplitudes are similar to bow echo. From these images, we label $1,148$ bow echo samples and 443 non-bow echo samples.

B. Prototype Extraction

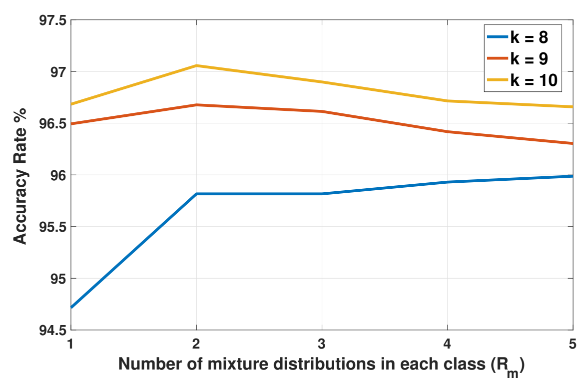
As it was explained in section III-C, we need to implement k-medoids algorithm [24] to find k different bow echoes as prototypes of bow echo class. Consequently, we first find the distance between all bow echoes by running shape matching algorithm pairwise on all bow echo class samples. After finding distance matrix between all pairs of bow echoes, by running k-medoids algorithm, k bow echo prototypes are extracted, as it is depicted in Figure 6, for $k = 8$.

C. Classification

The last step would be the training of the MDA classifier and classification process. For this step, we use cross-out validation to get more accurate results on our database. 20-fold cross-out validation would chunk data to 20 parts, each of which containing both bow echo and non-bow echo samples. In each iteration, one chunk of data would be considered as *test* data, and the other 19 chunks of data would be used as *training* data. The results of using MDA algorithm with 8, 9, and 10 prototypes ($k = 9, 8, 10$) is shown in Figure 7b, which reveals that the best option for number of distributions is $R_m = 2$. The effect of the number of bow echo prototypes is computed in Figure 7a. Results, indicate that overall it has increasing order with adding more prototypes, but it became approximately constant after 5 prototypes. Since for finding bow echo prototypes, the k-medoids algorithm should run separately each time, these prototypes could be completely different. For instance, for $k = 3$ and $k = 4$, the k-medoids



(a)



(b)

Figure 7: Accuracy Rate versus (a) number of bow echo prototypes, and (b) number of mixture distributions in each class.

algorithm can choose totally distinct prototypes. Thus, as our proposed algorithm depends on topological features of these distinct prototypes, the results of classification can vary based on these features. To compensate this effect, we apply moving average on the results, which makes it more reliable and robust against randomness inherent in choosing prototypes. We can go beyond ten prototypes to get better results, but the disadvantages of increasing computational time by adding more prototypes is higher than the benefits of small portion of improvement in accuracy rate.

V. Conclusions and Future Work

In this paper, we presented a novel computational approach for detecting bow echo patterns in radar images. Meteorologists can use our method to record bow echoes automatically and with high accuracy. In addition to detection, the next step in this research would be forecasting of bow echoes and severe weather conditions associated with them. This could be possible in this framework as well. With the help of affine indicator introduced in Section III-A, we can track changes in a line over some periods of time to see whether it is going to deform to a bow echo shape or not. Hence, this framework could be exploited for further developing of forecasting techniques.

Beyond the meteorological point of view, this research suggests more opportunities for future research direction. Shape matching is one of the most challenging areas in computer vision from early stages of formation of the field. This research suggests a new and robust descriptors for shapes to be used in shape matching context. On the other side, the innovative approach for skeletonization and skeleton pruning using fuzzy logic can be used in other schemes in image processing and computer vision for extracting well-structured skeletons in close proximity to human vision.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1027854. The authors would like to thank Jia Li of Penn State University and Michael A. Steinberg of Accuweather Inc. for helpful discussions, the National Oceanic and Atmospheric Administration (NOAA) for providing the data, and Yu Zhang and Yukun Chen for assisting with data collection.

References

[1] B. A. Klimowski, M. R. Hjelmfelt, and M. J. Bunkers, "Radar observations of the early evolution of bow echoes," *Weather and forecasting*, vol. 19, no. 4, pp. 727–734, 2004.

[2] B. A. Klimowski, M. J. Bunkers, M. R. Hjelmfelt, and J. N. Covert, "Severe convective windstorms over the northern high plains of the united states," *Weather and forecasting*, vol. 18, no. 3, pp. 502–519, 2003.

[3] C. Davis, N. Atkins, D. Bartels, L. Bosart *et al.*, "The bow echo and mcv experiment," *Bulletin of the American Meteorological Society*, vol. 85, no. 8, p. 1075, 2004.

[4] R. W. Przybylinski, "The bow echo: Observations, numerical simulations, and severe weather detection methods," *Weather and forecasting*, vol. 10, no. 2, pp. 203–218, 1995.

[5] T. T. Fujita, *Manual of downburst identification for project NIMROD*. Satellite and Mesometeorology Research Project, Department of the Geophysical Sciences, University of Chicago, 1978.

[6] Iowa Environmental Mesonet, "Documentation on IEM generated nexrad composites," 2001. [Online]. Available: http://mesonet.agron.iastate.edu/docs/nexrad_composites/

[7] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson, "Skeleton based shape matching and retrieval," in *Shape Modeling International*, May 2003, pp. 130–139.

[8] R. Ogniewicz and M. Ilg, "Voronoi skeletons: Theory and applications," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1992, pp. 63–69.

[9] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker, "Shock graphs and shape matching," *International Journal of Computer Vision*, vol. 35, pp. 13–32, 1999.

[10] B. Kégl and A. Krzyzak, "Piecewise linear skeletonization using principal curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 59–74, 2002.

[11] P. Dimitrov, C. Phillips, and K. Siddiqi, "Robust and efficient skeletal graphs," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1. IEEE, 2000, pp. 417–423.

[12] F. Reinders, M. E. Jacobson, and F. H. Post, "Skeleton graph generation for feature shape description," in *Data Visualization*. Springer, 2000, pp. 73–82.

[13] H. Blum, "Biological shape and visual science (part i)," *Journal of Theoretical Biology*, vol. 38, no. 2, pp. 208–287, 1973.

[14] X. Bai, L. J. Latecki, and W.-Y. Liu, "Skeleton pruning by contour partitioning with discrete curve evolution," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 3, pp. 449–462, 2007.

[15] H. Liu, Z.-H. Wu, X. Zhang, and D. F. Hsu, "A skeleton pruning algorithm based on information fusion," *Pattern Recognition Letters*, vol. 34, no. 10, pp. 1138–1145, 2013.

[16] W. Shen, X. Bai, X. Yang, and L. J. Latecki, "Skeleton pruning as trade-off between skeleton simplicity and reconstruction error," *Science China Information Sciences*, vol. 56, no. 4, pp. 1–14, 2013.

[17] L. A. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.

[18] J. Kacprzyk, *Studies in Fuzziness and Soft Computing*. Springer, 2000.

[19] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[20] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1982.

[21] L. Kaufman and P. Rousseeuw, *Clustering by means of medoids*. North-Holland, 1987.

[22] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics Springer, Berlin, 2001, vol. 1.

[23] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[24] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.