# Finding Structures in Weather Maps*

**Xingang Huang** **Feng Zhao**
Dept. of Computer and Information Science
The Ohio State University
2015 Neil Avenue
Columbus, Ohio 43210
{huang,fz}@cis.ohio-state.edu

**Abstract**

We describe an effective computational method for extracting structural information from spatial data sets. For instance, the structural information such as the location, shape, size, and movement about pressure cells and troughs are important for weather prediction. We are interested in extracting these structures and representing them at aggregate levels where salient geometric features are highlighted. The computational challenge is to quantify these features at multiple levels of granularity and find them incrementally. We introduce two types of spatial adjacency relations for spatial objects and present an effective algorithm for extracting and abstracting structural objects using the relational information. An application to finding local pressure cells in the weather data sets is demonstrated.

**Keywords.** Spatial knowledge discovery, proximity relationships, weather map analysis, spatial reasoning, computational tools and algorithms.

*We thank Chris Bailey-Kellogg for constructive comments on an early version of the paper, and Chris and Ivan Ordonez for providing the basic SAL computational environment to experiment with many of the ideas developed in this paper. The work is supported in part by NSF NYI grant CCR-9457802, ONR YI grant N00014-97-1-0599, and a Xerox grant to the Ohio State University.

# 1 Introduction

Because of the rapid advances in computational hardwares in recent years, numerical simulation plays an increasingly important role in weather forecasting. However, for the most part meteorological data interpretation still remains as a task for trained experts. In a typical weather map, information such as high/low pressure centers, pressure troughs, and warm/cold fronts is implicit but plays a crucial role in weather prediction. These features are often geometric, spatially distributed, and similar to those studied in computer vision. The major challenge to automate the data interpretation process is to identify and extract these salient features in weather maps and organize them into a coherent structure that can be efficiently queried for further processing.

This paper introduces an effective computational mechanism for extracting structural hierarchies from spatial relations in data sets such as weather maps. It is built upon the framework of Spatial Aggregation(SA) that features a multi-layer representation and a set of generic operators to transform the representation [11], and it significantly extends the framework to address how higher-level spatial relations are built from more primitive ones. We choose weather map analysis as our problem domain because of its richness in spatial information. Most weather features are ill-defined. Building a theory about spatial feature formation and a computational mechanism for extracting these features will have a significant impact on a range of spatial data mining problems. The work reported here represents a significant first step towards this ultimate goal, and is part of a larger project investigating spatial reasoning and data analysis [13].

# 2 Weather Map Analysis

Weather analysis is a well-studied subject. Given a weather map such as an iso-bar or iso-thermal chart, wind streamlines, or density distribution, experts describe weather features as geometric objects, along with their spatial distribution and movement across space and over time. Rules of thumb are often used to correlate different weather features and establish prediction patterns. Here is a sample of such descriptions [6]:

- "At 850mb, the polar front is located parallel to and on the warm side of the thermal packing."
- "Identify 700mb trough and ridge positions to establish vertical consistency (stacking) with surface and 850mb features."
- "Major and minor 500mb troughs are good indicators of existing or potential adverse weather."
- "Lows tend to stack toward colder air aloft while highs tend to stack toward warmer air aloft."

Figure 1 shows some of the features mentioned in the above description. Although these weather features are well understood once shown graphically, many of them are ill-defined and hence difficult to quantify and extract using brute-force numerical methods. One of the

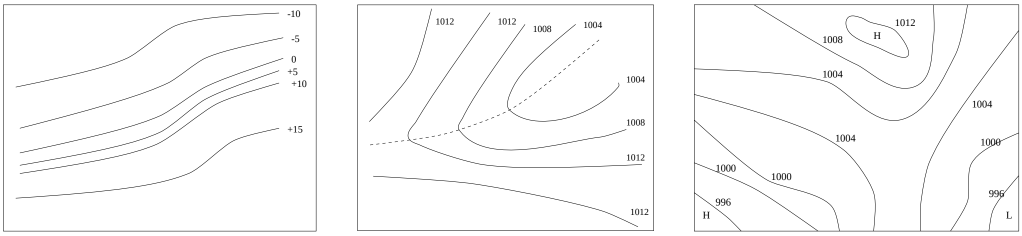



Figure 1: Weather Features: Left — thermal packing; Middle — pressure troughs, where solid lines representing iso-bars and dashed line trough position; Right — highs "H" and lows "L" in a pressure map.

objectives of this paper is to introduce spatial relations upon which these features can be defined and extracted. We are also interested in developing a computational environment in which data analysis programs can be prototyped using operators we introduce.

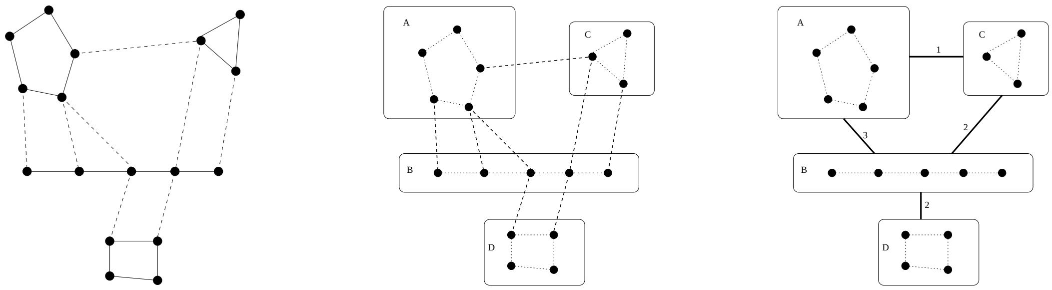
# 3 Finding Structures in Spatial Data

In this section, we introduce two types of neighborhood relations among spatial objects, *strong-adjacency* and *weak-adjacency*. The spatial structures are defined hierarchically using the adjacency relations. We then develop operators that incrementally extract the structural information, as an extension to Spatial Aggregation.

## 3.1 Spatial Aggregation

Spatial Aggregation(SA) is a framework for hierarchical data analysis tasks. It computes a multi-layer of intermediate representations, called spatial aggregates, by forming equivalence classes and adjacency relations. It uses a data structure, the neighborhood graph, as a common interface to modularize computations. A set of lower-level objects can be aggregated into a higher-level object if they share similarities. The input to SA is a numerical field, a mapping from one continuum(say $R^m$) to another (say $R^n$)) [2, 11]. For instance, a planar pressure field maps $R^2$ to $R^1$.

SA has been successfully applied to several difficult problems such as control and interpretation of numerical experiments [8], kinematics analysis of mechanisms [3], design of controllers [12], reasoning about fluid motion [9, 10], and distributed control optimization [1]. In these problems, the structures in input data sets are induced from equivalence and neighborhood relations. In this paper, we refine the neighborhood relation in order to automate the process of building higher-level adjacency from lower-level ones and introduce additional operators to SA to support programming in this style for spatial data analysis problems. This work is largely inspired by Yip's pioneering work in structural inferences from massive fluid motion data sets [10].





(a) (b) (c)

Figure 2: Spatial adjacencies and their aggregation: (a) Spatial objects and neighborhood relations. Solid lines as strong-adjacency and dashed lines as weak-adjacency. (b) Aggregate objects A, B, C, and D are formed using strong-adjacency relations that become internal to the aggregate objects. The weak-adjacency now reveals inter-object relations among the aggregate objects. (c) The neighborhood relations among the aggregate objects are constructed from aggregating weak-adjacencies. The number next to each relation is its weight, computed from counting the number of weak-adjacencies from which the relation is built.

## 3.2 Spatial adjacency

Spatial objects, when embedded in a metric space, relate to each other via adjacency relations. These objects form clusters and organize themselves into hierarchies in many application domains such as weather maps. For example, nearby iso-thermals group to form a thermal packing, as shown in Fig. 1(left). We distinguish two types of adjacency relations. The strong-adjacencies bond tightly coupled objects together to form the next higher-level object. The weak-adjacencies, on the other hand, are the basis for forming adjacencies among the aggregate objects (Fig. 2(a)).

### 3.2.1 Aggregate objects from strong-adjacency

At the base-level of aggregation, a user-provided predicate *is-strong-adjacency* classifies neighborhood relations into strong and weak adjacencies. The classification depends on task-specific requirements and domains. The strong-adjacencies are used transitively to generate an equivalent relation which we use to partition spatial objects into equivalence classes. Each equivalence class with strong adjacencies among its members will then be re-described into the next higher-level spatial object (Fig. 2(b)).

The internal structure of an aggregate object defines how the object is formed using the strong-adjacency relations. In addition, the spatial object may have other more application-specific structures defined on its constituent objects. For example, a linear spatial object may have *previous* and *next* relations defined among its constituents so that an linear ordering can be established.

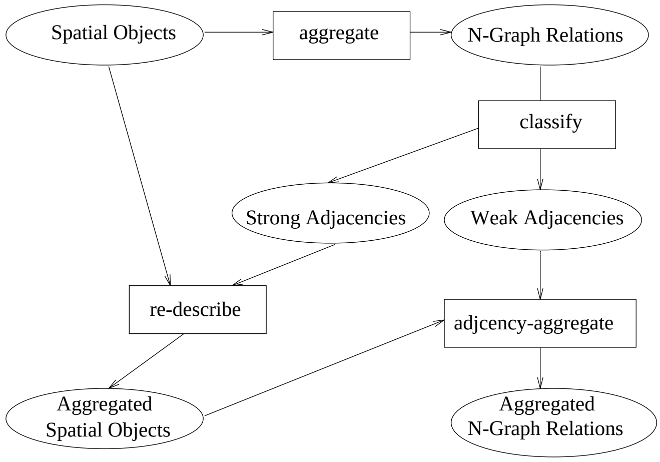



Figure 3: The structure finding mechanism for one layer of aggregation. The output is fed into the next layer of aggregation that has the identical computational structure. Rectangles denote operators and ovals data.

### 3.2.2 Neighborhood relation from weak-adjacency

The weak-adjacencies aggregate to form neighborhood relations among aggregate objects (Fig. 2(c)). We assign a weight to each aggregate neighborhood relation to denote the number of lower-level weak-adjacencies that the neighborhood relation is built from. The weight can serve as an indicator for the strength of the neighborhood relation and help determine which is a strong adjacency at the aggregate level.

The operator for aggregating weak-adjacencies is *adjacency-aggregate*. It returns a neighborhood graph for the aggregate objects: if two aggregate objects relate to each other via a weak-adjacency, establish an edge in the graph. This approach is particularly useful when the objects have sufficiently rich internal structures, while abstracting them as points in an Euclidean space does not suffice.

### 3.2.3 A structure finding algorithm

Figure 3 summarizes an algorithm for incrementally finding structural relations among spatial objects. At each level, neighborhood relations are aggregated into adjacencies internal to aggregate objects and more abstract neighborhood relations. At the next level, with the internal structure of objects abstracted away and the relations between objects simplified by aggregation, the amount of information is greatly reduced to facilitate extraction of qualitative patterns. When object details become necessary, the internal structure of the object is available so that relevant information can be quickly located. In summary, the algorithm is capable of not only explicating structures from data, but organizing information in a structured, hierarchical way. This makes the approach particularly suitable for applications with data sets containing huge amounts of implicit information such as spatial data.

### 3.3 Programming Support

To support programming for structure finding applications, we have defined and implemented several commonly used data types and operators, extended from the original C++ SA library [2]. Table 1 summarizes the main features:

- A spatial objects is required to have the basic structure — strong-adjacencies among its constituents.
- Neighborhood graph (NGraph) is represented as a collection of binary relations.
- A new operator — **adjacency-aggregate** is introduced for aggregating weak-adjacencies.
- The operators **classify** and **re-describe** are modified to work with strong-adjacencies and weak-adjacencies.

## 4 An Illustration: labeling pressure cells in weather maps

In this section we will show how the Structure Finding algorithm labels high/low pressure center areas in a weather map.

### 4.1 Iso-curve neighborhood graph and local extrema

The labeling problem here is to find well-formed "centers" that are also local extrema from an iso-bar data set. To begin, we first extract iso-curves from the data set, using a contouring algorithm to be detailed later. To determine if an iso-curve is a local extremum, we need to establish a neighborhood graph for the iso-curve objects in a chart.

Each iso-curve partitions the entire chart into two disconnected regions. Topologically, we define two curves to be neighbors if they are not separated by another curve. We call this neighborhood graph the topological neighborhood graph for iso-curves. Figure 4 gives an example of a weather chart and its topological neighborhood graph for iso-curves.

In Figure 4(b), there are five extremal nodes: (a,3), (b,2), (d,2), (j,6) and (g,1). However, from Figure 4(a) it is clear that (b,2) and (d,2) do not form centers in the region. The curves (a,3), (j,6) and (g,1) do form centers, and they are non-articulation nodes in the topological neighborhood graph[1].

The labeling algorithm identifies iso-curves that are both extrema and non-articulation nodes in the topological neighborhood graph as the "centers".

### 4.2 A labeling algorithm

The program takes a pressure data field as input. It performs contouring to generate iso-curves and builds a neighborhood graph for the iso-curves. The final output is a contoured weather chart with high/low center areas correctly labeled.

The algorithm works as follows:

[1] An articulation node partitions a graph into two disconnected components when removed.

- Data types:
  - Spatial Object:
    - **is-constituent** : object $\rightarrow$ boolean
      Return true if the object is one of the constituent spatial objects.
    - **is-joined-by-strong-adjacency** : object * object $\rightarrow$ boolean
      Return true if two constituent spatial objects are joined by a strong-adjacency.
  - Relation: a three tuple (object, object, weight).
  - NGraph: a set of relations with the methods:
    - **neighbors** : object $\rightarrow$ objects
      Return the neighbors of a given object.
    - **neighbor-relations** : object $\rightarrow$ relations
      Return all the neighborhood relations of an given object.
    - **is-neighbor**: object * object $\rightarrow$ boolean
      Return true if the two objects are neighbors in the Ngraph.

  Strong-adjacencies and weak-adjacencies are also implemented in NGraphs since they are subsets of the relations in a neighborhood graph.

- Operators:
  - **aggregate** : objects * ngraph-fac $\rightarrow$ Ngraph
    Groups objects into an Ngraph, according to a constructor ngraph-fac.
  - **classify** : ngraph * is-strong-adjacency $\rightarrow$ strong-adjacencies * weak-adjacencies
    Classify the neighborhood relations into strong-adjacencies and weak-adjacencies using the user provided predicate 'is-strong-adjacency'.
  - **re-describe** : objects * strong-adjacencies * redescribe-objs-op $\rightarrow$ objects
    Redescribe a collection of spatial objects into higher level spatial objects using strong-adjacencies and a user provided function 'redescribe-objs-op'.
  - **adjacency-aggregate**: objects * weak-adjacencies * redescribe-adjs-op $\rightarrow$ ngraph
    Aggregate weak-adjacencies into higher level Ngraph relations between aggregated spatial objects using a user-provided function 'redescribe-adjs-op'.

Table 1: Data types and operators in the SA Structure Finding library.

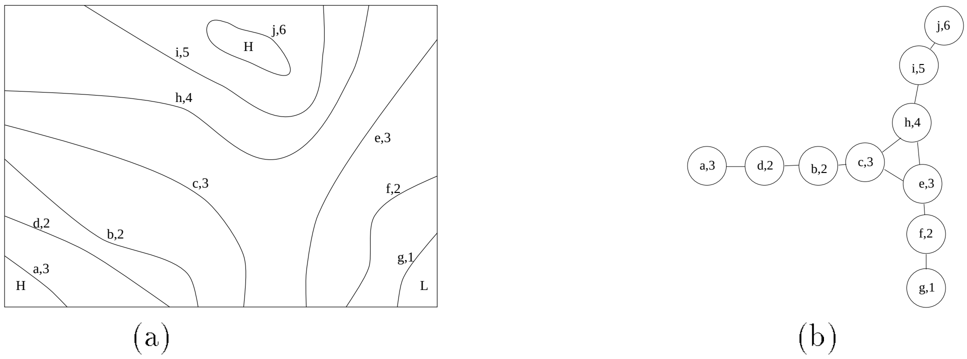



Figure 4: (a) A weather chart consisting of a group of iso-curves, each of which is identified by a pair (identifier, iso-value). The centers are labeled “H” or “L”. (b) A topological neighborhood graph for the iso-curves.

1. Preprocess the data field and extract all the iso-points of required contour values. (Fig 5(a)).

2. Aggregate all the iso-points and build a delaunay neighborhood graph on them. (Fig 5(b))

3. Classify the iso-point neighborhood relations into strong-adjacencies (Fig 5(c)) and weak-adjacencies (Fig 5(d)). The *is-strong-adjacency* predicate takes a neighborhood relation as the input and returns true if the two iso-points are of the same value and their distance is within a threshold.

4. Re-describe iso-points and their strong adjacencies into iso-curves. This completes the contouring part of the algorithm.

5. Aggregate weak-adjacencies to obtain a neighborhood graph for the iso-curves (Fig 5(g)).

6. Detect all the local extrema and non-articulation nodes in the iso-curve neighborhood graph and label them as high/low center areas and output the final result (Fig 5(h)).

Notice that the neighborhood graph (Figure 5(g)) computed using adjacency-aggregation is a subset of of the topological neighborhood graph. This reduced neighborhood graph could potentially let the labeling algorithm miss certain high/low value centers. However, this does not turn out to be a problem for real world weather data sets; the missed centers in pathological cases correspond to those that are not well formed in the first place. Tests on various weather data sets have shown that the algorithm is robust.

The program is written in C++ using the SA Structure Finding Library. It runs reasonably efficient, taking about 10 seconds to process a 80×40 weather data set on a 200MHz Pentium NT machine.

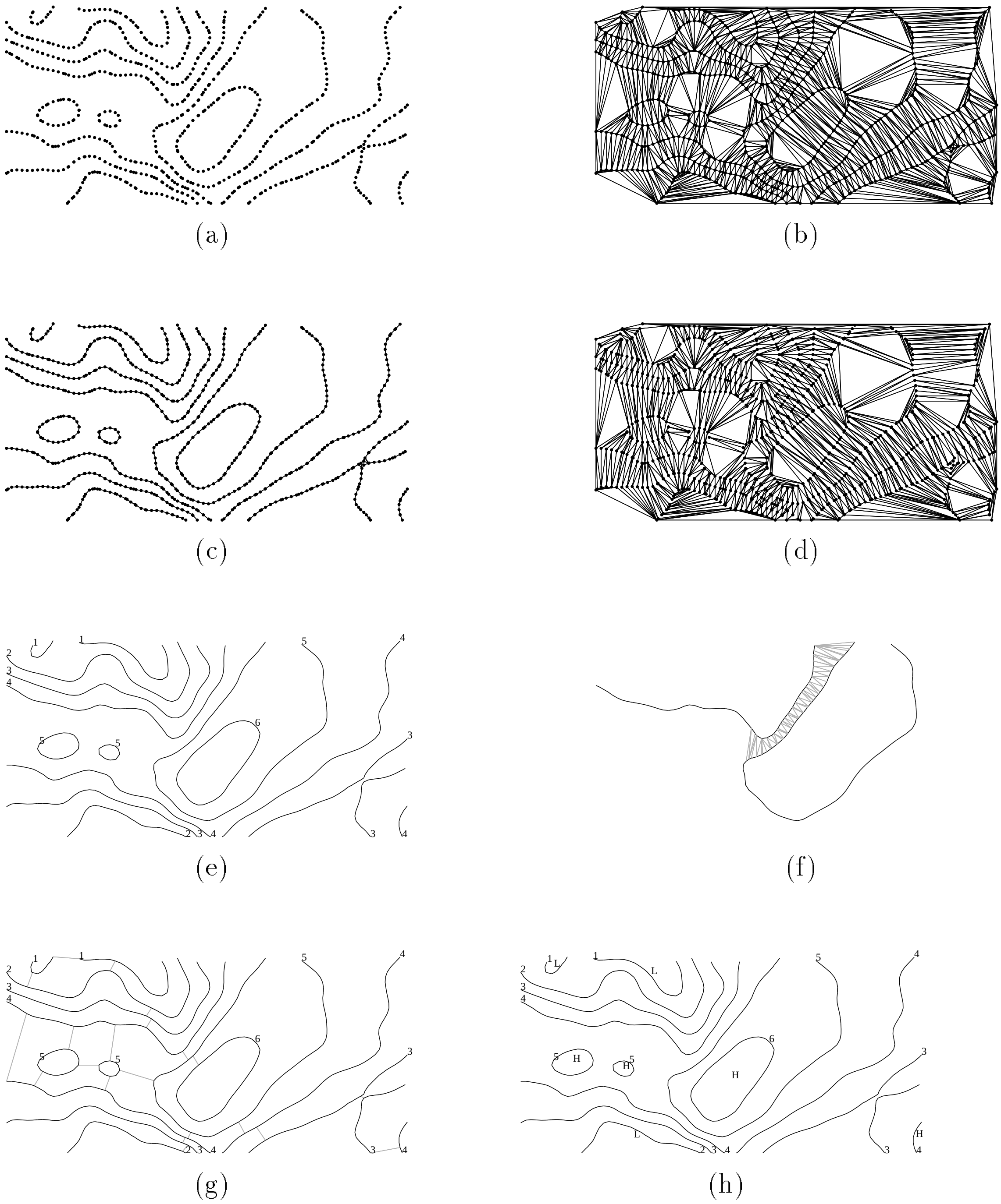



Figure 5: (a) Iso-points, computed from input pressure maps. (b) Neighborhood relations, built from delaunay graph, a commonly used neighborhood graph. (c) Strong adjacencies among iso-points. (d) Weak adjacencies among iso-points. (e) Iso-curves, aggregated from iso-points and their strong-adjacencies. (f) Weak-adjacencies between the constituent points of two iso-curves. (g) Neighborhood relations between iso-curves, aggregated from lower level weak-adjacencies. (h) Output — high/low pressure areas labeled as “H” and “L” respectively.

# 5 Related Work and Discussion

The area of spatial data mining and knowledge discovery has been very active. Leading researchers have developed many efficient algorithms for different types of spatial data sets. For instance, Lu et al. [7] developed the SD/NSD algorithms to extract high-level relationships from spatial and nonspatial data. The algorithms assume that a spatial hierarchy is given a priori. Ng and Han [5] described a very efficient algorithm, CLARANS, to find spatial clusters of points in large databases. Knorr and Ng [4] presented the CRH algorithm for building proximity relations among aggregated spatial structures, represented as point clusters, and features, represented as polygons.

The main feature of the structure finding algorithm described in this paper is that it builds and maintains intermediate spatial structures so that further processing can query the structure and resolve inconsistencies when additional information becomes available. In the weather problem, the algorithm not only detects the high/low value centers, but also builds iso-curve neighborhood graph and maintains a linear structure internal to iso-curve objects and weak-adjacencies between iso-curves. These intermediate structures are extremely useful in extracting additional information such as point-to-curve and curve-to-curve distances and for reasoning about relative spatial distribution of objects. On the other hand, the method of [4] could potentially help us correlate heterogeneous spatial features across different weather maps.

The labeling algorithm presented here is quite robust in finding local extremal points. In contrast, a direct sampling algorithm for finding local extrema might suffer from noise; some smoothing of the data is necessary.

# 6 Conclusion

This paper has developed a computational mechanism for identifying and extracting structural information from spatial data sets. In this mechanism, neighborhood relations are classified into strong-adjacencies and weak-adjacencies according to the roles they play in building higher-level aggregate objects. The weak-adjacencies are aggregated into neighborhood relations at the next level for building more global structures.

The contribution of this work is two-fold: First, it provides a systematic approach to finding structural information from the spatial data by identifying adjacency relations and aggregation operators; Second, it organizes the spatial information in a structured, hierarchical way, which is essential for efficiently extracting information at multiple levels of details and for mining spatial data sets containing large amounts of implicit information.

The structure finding mechanism is illustrated in labeling high/low pressure centers in weather maps, with encouraging performance results. Future work involves extracting other features, such as pressure trough, jet stream, and fronts in weather maps. Features such as fronts can be extracted by correlation across time. The temporal effect as well as the necessity to correlate different fields (e.g., pressure, temperature, humidity) are some of the other challenges.

# References

[1] C. Bailey-Kellogg and F. Zhao. Spatial aggregation: modeling and controlling physical fields. In *Proc. 11th International Workshop on Qualitative Reasoning*, 1997.

[2] C. Bailey-Kellogg, F. Zhao, and K. Yip. Spatial aggregation: language and applications. In *Proceedings of AAAI*, 1996.

[3] L. Joskowicz and E. Sacks. Computational kinematics. *Artificial Intelligence*, 51:381–416, 1991.

[4] E.M. Knorr and R.T. Ng. Finding aggregate proximity relationships and commonalities in spatial data mining. *IEEE Trans. on Knowledge and Data Engineering*, 8:884–897, December 1996.

[5] R.T. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In *Proceedings of the Twentieth International Conference on Very Large Databases*, pages 144–155, Santiago, Chile, 1994.

[6] Air Weather Service. Back to basics. In *AWS FOT Seminar STT-Q9-0004*, 1975.

[7] J. Han W. Lu and B.C. Ooi. Discovery of General Knowledge in Large Spatial Databases. In *Proc. Far East Workshop on Geographic Information Systems*, pages 275–289, Singapore, 1993.

[8] K. M. Yip. *KAM: A system for intelligently guiding numerical experimentation by computer.* MIT Press, 1991.

[9] K. M. Yip. Reasoning about fluid motion: finding structures. In *Proceedings of IJCAI*, 1995.

[10] K. M. Yip. Structural inferences from massive datasets. In *Proceedings of IJCAI*, 1997.

[11] K. M. Yip and F. Zhao. Spatial aggregation: theory and applications. *J. Artificial Intelligence Research*, 5, 1996.

[12] F. Zhao. Extracting and representing qualitative behaviors of complex systems in phase spaces. *Artificial Intelligence*, 69(1-2):51–92, 1994.

[13] F. Zhao. Project web page, intelligent simulation group at ohio state. `http://www.cis.ohio-state.edu/insight`, 1997.