Thesis for the Degree of Doctor of Engineering

Fast Doppler Radar Image Generation System based on Tornado Recognition using Deep Learning

by

Liu Yang

Department of IT Convergence and Application Engineering

The Graduate School

Pukyong National University

June 2016

Fast Doppler Radar Image Generation System based on Tornado Recognition using Deep Learning

딥러닝을 이용한 토네이도 인식 기반 고속 도플러 레이더 영상 생성 시스템

Advisor: Prof. Ki-Ryong Kwon

by

Liu Yang

A thesis submitted in partial fulfillment of the requirements
for the degree of doctor of engineering

In Department of IT Convergence and Application Engineering, The Graduate School,
Pukyong National University

June 2016

Fast Doppler Radar Image Generation System based on Tornado Recognition using Deep Learning

A Thesis
by
Liu Yang

Approved by:

_____
(Chairman) Kwang-Seok Moon

_____
(Member) Suk-Hwan Lee

_____
(Member) Bong-Kee Sin

_____
(Member) Heug-Kook Choi

_____
(Member) Ki-Ryong Kwon

June 24, 2016

Contents

Table Contents

Figure Contents

딥러닝을 이용한 토네이도 인식 기반 고속 도플러 레이더 영상 생성 시스템

Liu Yang

부 경 대 학 교 대 학 원 IT 융 합 응 용 공 학 과

요 약

토네이도는 가장 강력하고 파괴적인 대기 현상이다. 일반적으로 구름에서 공기의 기둥이 형성되어 회전하는 것을 제시한다. 토네이도는 매년 수백만 달러의 피해와 수백명의 사상자를 발생시킨다. 최근 몇년동안, 인류는 관측 데이터 처리 및 정보 분석을 포함한 조밀 도플러 레이더에 기반한 심각한 기상 경고와 단시간 기상 예보를 구축하려고 한다. 주요 목표는 토네이도 탐지 및 예측이다. 하지만 빠른 기상 예보의 한 가지 문제점은 대규모 데이터 처리에 의해 제한된다는 것이다. 분석에서는 레이더 데이터 생성에서 많은 시간을 요구하는 데이터 보간을 보여준다. 또 다른 하나는 대한 전형적인 토네이도를 검출하는 기능이 불충분하다는 점이다. 검출 처리에서 CNNs(Convolutional Neural Networks) 사용과 MRF분할 기반 계층적 등고선을 이용하여 레이더 이미지를 생성하는 방법을 제안한다.

기존의 방법과 달리, 데이터가 없는 부분을 계산하기 위해 다양한 보간 방법을 사용하지만, 레이더 이미지의 반사율 규모의 각 층에서 윤곽을 검색하는 방법에 초점을 두지 않았다. 등고선검색 방법은 기존의 방법에서 질량 반복 보간 처리를 회피함으로써, 처리 시간을 향상시킬 수 있다. 본 논문에서는, 범용 포맷(UF) 원시 레이더 파일이 검색된다. 다른 방법에 비해, 제안하는 방법은 신뢰할 수 있는 품질을 보다 신속하게 레이더 이미지를 생성할 수 있다. 그러면 인류는 기상 재해 경고에 대한 더 많은 리드 타임을 얻을 수 있다.

토네이도 인식 처리에 있어, 하나 또는 여러 기능을 사용하는 기존의 방법과 비교하여, 레이더 이미지의 정보와 기능을 처리하는 CNNs

방법을 사용한다. CNNs는 더 나은 강인성과 정확성을 가지고 있다. 우리의 모델을 학습하기위해 레이더의 기본 속도 이미지를 사용한다. 모델은 CaffeNet을 기반으로 한다. 모델은 미세 조정을 통해 레이더 이미지을 매칭시킬 수 있다. 그리고 최종적으로 더 빠르고 더 나은 인식률을 제공한다.

Fast Doppler Radar Image Generation System based on Tornado Recognition using Deep Learning

Liu Yang

Department of IT Convergence and Application Engineering, The Graduate School, Pukyong National University

Abstract

Tornado is the most strong and violent atmospheric phenomena. Usually it presents like rotating column of air from cloud to the ground. Each year it results in hundreds of millions dollars damage and around hundreds fatalities. In recent years, people are attempting to build a system for short-time weather forecast and severe weather warning base on dense Doppler radars, including observation, data processing and information analysis. A major goal is tornado detection and prediction. But one problem is that the rapid weather forecast is limited by large scale data processing. Analysis shows that data interpolation costs a lot of time in radar data generation. Another one is that the conventional tornado features for detection is insufficient. We propose a method to faster generate radar image by using hierarchical contour-line based on MRF segmentation and use CNNs(Convolutional Neural Networks) in our detecting process.

Unlike conventional methods, we do not use various interpolation methods to calculate the dataless parts, but focus on how to retrieve the contour in each layer of reflectivity scale of radar image. Contour-line retrieval method can improve the processing time by avoiding the mass repetitive interpolation process in conventional methods. In this paper, the Universal Format (UF) raw radar files is retrieved. Compared with other methods, our method can generate radar image more rapidly with reliable quality. Then people can gain more lead time for

weather disaster warning.

In tornado recognition process, we use CNNs method to process the features and information of radar image, comparing with conventional methods which only use some individual one or several features. CNNs has better robustness and accuracy. We use base velocity images of radar as input to train our model. The model is based on CaffeNet. Through fine-tuning the model can suit our radar images and finally gives a faster and good recognition rate.

Ⅰ. Introduction

1.1 Development of short-range dense Doppler radar network

Weather information is an important part of daily life. A global data (82 countries) from EMDAT (Emergency Events Database) shows that there are 10912 major nature disasters between 1970 and 2013, the disasters which are related to the meteorological reach 63 percentage like cyclone, storm and flood etc... Economic losses from meteorological disasters reach an average of more than 2 hundred billion each year. More than one million people are dead and almost 5 billion are affected in these 43 years [1]. Rapid weather forecast can help people to reduce weather-related losses and enhance societal benefits including property and life.

People have researched meteorology for decades, radar has been initially introduced into meteorological fields since last century 50s. People learned that besides monitoring enemy aircraft, radar also worked well at certain wavelengths (about 3 to 10 centimeters) for detecting precipitation. Then people find an effective way to study and track precipitation. Introducing Doppler radar for weather forecast becomes Next great advance. Comparing with conventional radar, Doppler radar can not only detect the precipitation, by also produce velocity data by using the Doppler Effect. Doppler radar is commonly used as weather radar since the Weather Surveillance Radar - 1988 Doppler radar in United States, which is belong to NEXRAD (Next Generation Weather Radar). These radars locate in different places, provide the speed and direction data of precipitation, then make it possible to warn and forecast some severe weather like thunderstorms and hurricane etc.

Despite significant performance and capability, the ability of long-range radar is impeded for detecting and identify small tornadoes and other finescale weather phenomena [2]. Taking the WSR-88D as an example, the radar can cover an area of 230~345 km radius, but more than 70% of the troposphere below 1-km altitude above ground level (AGL) cannot be observed [3]. Due to Earth's curvature and terrain blockage, one major problem with the long-range radars is that they are not able to observe the lower parts of the atmosphere adequately. A study from published research found that during 2008 approximately 75% of tornado warnings issued by the National Weather Service (NWS) were false alarms which is with 72% detection probability and 13-min lead time [4]. Apparently, this cannot satisfy people's need. Another major problem of radar is the spatial resolution at the far end. Radar spreads microwave to the apace around radar station, the resolution of radar contains two dimensions: radial resolution and azimuthal resolution. Radial resolution is a certain data, it is the number of bins for a given distance that radar is able to detect. The azimuthal resolution also call beamwidth, it is the number of radials that the radar can depict in scanning space of radar. From figure 1, we can find that the distance between two adjacent radials will increase with beam spreading, the gate of outer circle correspond to a larger area than the inner one. Then the resolution on the outside will be lower than the inside. Besides two previous points, the long-range radar also has some other weaknesses like the high frequency attenuation, high construction and operating cost etc… As solution, a project named CASA is developed to overcome these limitations.

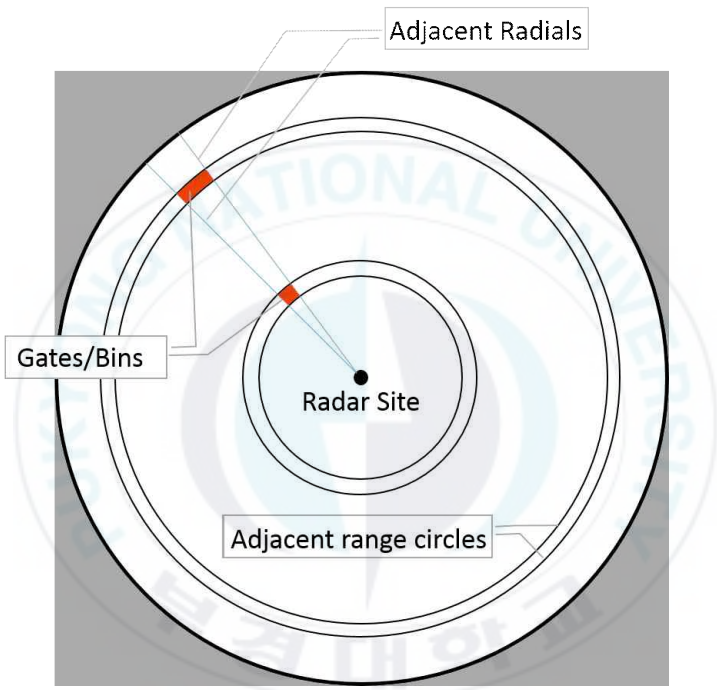
For overcoming the limitations of long-range radar and continually improving the capability of weather radar, National Science Foundation (NSF) Engineering Research Enter (ERC) developed next generation weather-sensing networks, Collaborative Adaptive Sensing of the

Atmosphere (CASA). In contrast to the long-range radar, CASA tries to employ a kind of low-cost, dense networks based on many short-range X-band (3.2 cm wavelength) Doppler radars. Taking the first test bed IP1 in Oklahoma United States as an example, the radar sites locate only few tens of kilometers apart from each other, each radar can detect an area of 40 km radius. This dense network has higher resolution and can see the lower troposphere better. Meanwhile short-range radars install easier and cost less. As CASA system makes the performance of weather forecast better. Timeliness problem becomes more and more conspicuous. Because CASA uses short-range radar, that means a very short period of warning time from detecting moment. Furthermore the dense networks means that more radar data need to be processed. So we need a faster method to retrieve radar data.

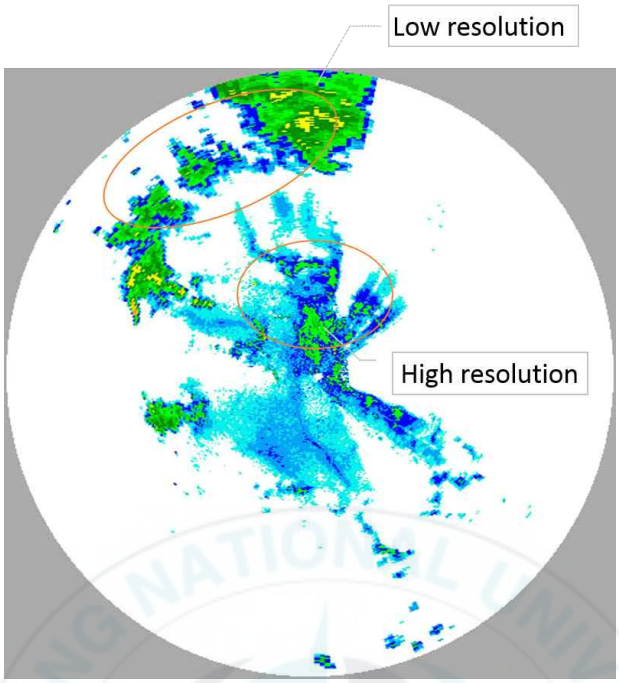
In radar data retrieval process, we need to calculate and estimate a lot of dataless parts in radar image generating step. Through analyzing the time distribution in many radar data retrieval process. We find that the most of retrieval time is spent on data interpolation step which is just a mass of repetitive computation. People developed many algorithms for gaining a fast interpolation method which also have an acceptable image quality. But even the simplest and fast one, it is still unsatisfactory. In our proposed method, we does not focus on the algorithm of interpolation. Paper proposes a radar retrieval method by hierarchical contour-line method, and use MRF to estimate and refine the contour. The major information of radar image can be describe by the contour from different layers, this will avoid a lot of repetitive calculation in radar retrieving parts. Then we use flood fill and image merging to integrate a final radar image. In experiment we retrieve UF raw data files by proposed method and compare with a conventional method. The results show that the proposed method can effectively reduce processing time through avoiding

the mass calculus of interpolation as expected.

This paper discusses how to fast generate weather radar data, then gain more warning lead time for preparation. Although we only talk the radar imaging, more generally the method can be used for hierarchical imaging in more fields.



(a)



(b)

Fig. 1. (a) The diagram of beam spreading of radar, (b) an example of radar image.

1.2 Low atmospheric layer detection with CASA network

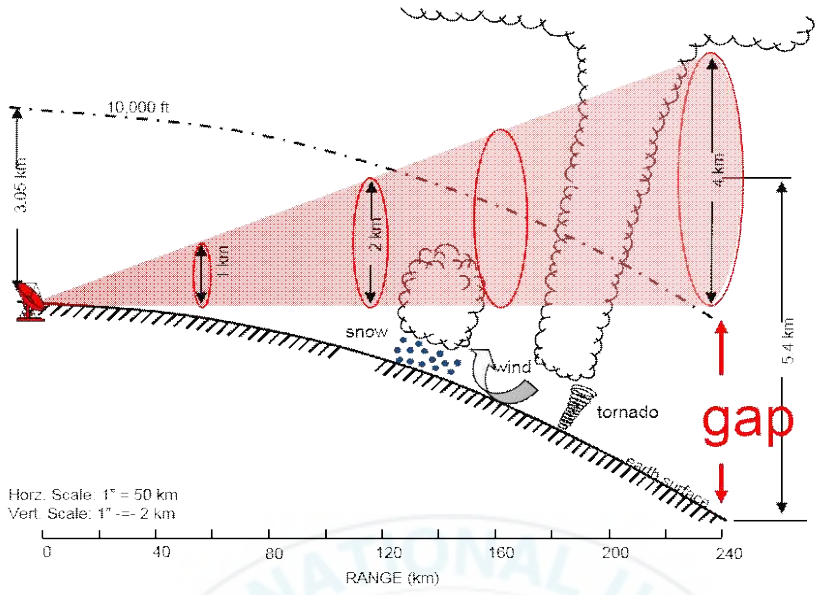
Tornadoes are the most eye-catching atmospheric phenomena, because they are rarely seen and its devastating impact. Tornadoes occur at any time of the year and in many place of the world, like America, Europe, Asia, Australia, and three highest concentrations are the United States, Argentina and Bangladesh. About 1,200 tornadoes pass over the United States each year, and most of them occur in south central of the United States, which is called "Tornado Alley"[5][6].

The strength of tornadoes can described by the EF-Scale (Enhanced Fujita scale) like table 1. Despite there are around one thousand tornadoes occur, but only about 2% of them reach F4 of F5 level on EF-Scale. Majority of tornadoes can not be detected, and hardly predict. The tornadoes occurring in rural place have little influence to people, but some may pass through urban localities. As previous mentioned, tornadoes only occur below 1-km altitude above ground level, it a blind zone to most weather radars (figure 2. (a)), even radar can get tornadoes, it is always the top part of tornadoes, and the rotation features of this part is not significant comparing with the lower part.

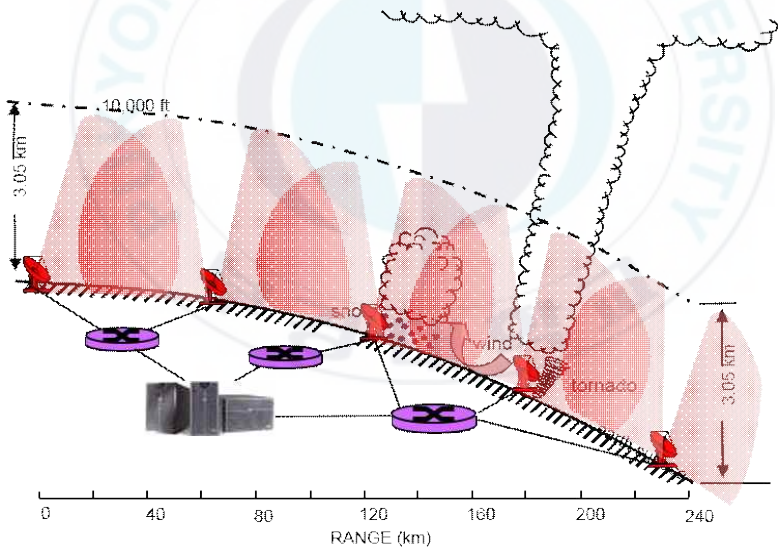
Comparing with other weather phenomena, tornado is always a small scale one, and also short-time. According to statistics, 69% of all tornadoes have lifetime of 1~10 minutes, 29% of all tornadoes last around 20 minutes and only 2% of all tornadoes can last over an hour. Such short lifetime brings us a problem it is difficult to track and research. So we need our radars have high spatial temporal resolution to resolve these signature (figure 2. (b)).

Table 1. Enhanced Fujita scale

Scale	Wind speed (km/h)	Potential damage
EF0	105-137	light
EF1	138-177	moderate
EF2	178-217	considerable
EF3	218-266	severe
EF4	267-322	devastating
EF5	>322	incredible



(a)



(b)

Fig. 2 (a) Conventional Doppler radar, (b) CASA short range Doppler radar system.

1.3 The analysis of features in machine learning

machine learning is a subject which focuses on how to simulate or learn people's behaviour and get new knowledge, meanwhile continuously reorganize excising knowledge to improve self-performance. The application fields of machine learning is very extensive, like image recognition, speech recognition, weather prediction, genetic expression etc. Machine learning made the breakthrough in 2006, on the hardware side, it depends on computer performance improvement and parallel processing capabilities to big data. On software side, it depends on a new algorithm which is deep learning[7]. With deep learning, people finally find a way to process abstract conception of objects.

Before machine learning, when people do objects detection, classification or recognition, the train of thought is data getting from sensor (like CMOS), pre-processing, feature extraction, feature selection then inference, prediction or recognition. During this procedure, the feature selection is a key part, it directly determine the result. Until now there are many effective features like SIFT, Hog, Textons, RIFT, GLOH etc. But feature selection is a laborsome thing, need much test, comparison and analysis. All of them have specific processing objects, non of them is all-purpose. Then the most important part of all procedure actually depends on people's knowledge, experience even luck.

A most important feature of deep learning is to learn feature automatically, so deep learning has another name, unsupervised feature learning[8]. Deep learning learns not only low-level features, but also structure features[9]. For instant, the face recognition process, to an image of face the feature on pixel-level is not valuable, the deeper level is edge feature. We know that image can be described by edge's combination

[10][11], on this level it is hard to tell what the difference between the edge of face and the edge of other object like a car (figure 3). A deeper level of face than edge feature can be like a series of patches of our eyes, nose and mouth, but you still can not confirm whether this is a face. Then a deeper level than these patch is our faces. The distribution of eyes, nose and mouth has structuredness, Using this structuredness we can confirm whether it is a face without doubt[12]. So when we want to express some more structure and complicated image, The expression on deeper level is necessary[13].

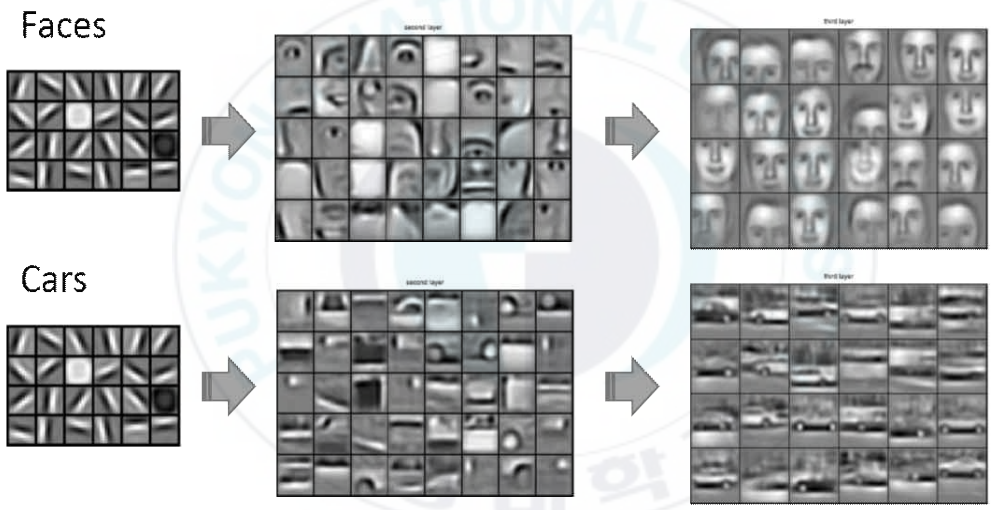


Fig. 3. Features of different objects from different level.

1.4 Outline of thesis

In following chapter 2 briefly introduces the related works about the conventional methods in radar imaging area, the CNNs method background and major algorithms.

Chapter 3 proposes our hierarchical contour-line retrieval method based on MRF segmentation, the content includes UF file reading, contour line abstraction, MRF segmentation. We shows the retrieval results by conventional method and proposed method, both performance and time distribution in different steps are also analyzed. In CNNs process we present the dataset which is used, and analyze the accuracy of the method

Chapter 4 We introduce the structure of CNNs method. Briefly talk about three major parts: convolutional layer, pooling layer and loss layer. Next step, we talk about the weight and loss calculation in forward and back propagation. In simulation part, we talk about the model design and tune. We use out data to analysis the results and the feature in tornadoes recognition.

Ⅱ. Related works

2.1 Reflectivity express in Doppler radar

Radar is coined as an acronym for “radio detection and ranging” by the United States Navy in 1940. This name accurately describes radar’s function. Radar emits electromagnetic energy from a transmitter into the atmosphere, some of the energy bounces back when it collides with some objects and radar will collect the reflected energy by its receiver. Then radar can detect these objects and measure their distance and size though analyzing the signal from the return energy. There are three major parts: 1) Radar Data Acquisition, 2) Radar Production Generation, and 3) User Display System [14]. In this process, two major parameters are produced: base reflectivity and velocity. In this paper, we discuss about radar data and reference algorithms and focus on how to generate radar image of these two parameters by using raw radar data.

Reflectivity displays the amount of reflected energy, which is used to establish the precipitation in atmosphere. A widely applied equation in meteorology the Probert-Jones radar reflectivity equation will help to quantify it,

$$P_r = \left[\frac{P_t G^2 \theta^2 H \pi^3 K^2 L}{1024(\ln 2)\lambda^2} \right] \times \frac{Z}{R^2} \qquad (1)$$

where P_r is the power returned to the radar from a target, P_t is the peak

transmitted power, G is the antenna gain, θ is the angular beamwidth, H is the pulse length, K is the physical constant depend on target character, L is the signal loss factors associated with attenuation and receiver detection, Z is the reflectivity, λ is the transmitted energy wavelength and R is the range from radar to target.

We can use a simplified radar equation to describe the relation between the reflectivity factor Z, the power returned by precipitation scatter P_r, the range from the target to the radar site L_α, attenuation factor and the radar constant C_γ.

$$Z = \frac{P_\gamma R^2}{C_\gamma L \alpha} \tag{2}$$

In practice the Z is a value which can range over many orders of magnitude, so people compress this large range of values for operational use, that is dBZ and the convert function is as following.

$$dBZ = 10\log_{10} Z \tag{3}$$

The usual range of reflectivity is between 0 dBZ and 80 dBZ, different value matches different precipitation category like table 2 [15]. In some of extreme cases the value can reach 90 dBZ, and reflectivity value also can be negative, for example a clear sky with a little dust or fog.

Table 2. Reflectivity threshold in different precipitation category.

Level	Reflectivity Interval dBZ	Precipitation description
1	18 - 30	Light precipitation
2	30 - 38	Light to moderate rain
3	38 - 44	Moderate to heavy rain
4	44 - 50	Heavy rain
5	50 - 57	Very heavy rain, hail possible
6	> 57	Very heavy rain and large hail

Velocity is a radar product that displays the wind speed. Radar can detect the wind speed is based on the Doppler Effect. The Doppler Effect is the change of wavelength caused by motion of the source. Because the spread of electromagnetic waves do not need medium, the shift in frequency only depend on the relative velocity. The Doppler Effect for electromagnetic waves is as following,

$$f' = f(1 \pm \frac{v}{c}) \tag{4}$$

where f is the observed frequency, f' is the emitted frequency and v is the relative velocity.

Normally in velocity image, red color means that winds are moving away from the radar, green color means that winds are moving towards the radar, and use darker color to denote slower winds, bright color to denote faster winds. Then in the velocity image, we can find a couple of reverse winds. If this couple of reverse winds was tight and bright enough on screen that means there could produce a tornado.

2.2 Coordinate mapping research in radar image retrieval model

After radar collects weather information and extracts raw radar data by signal processing, the further process is radar image generation for meteorological or other purposes. The most fundamental unit of radar data is gate or bin which is sampled from radar wave. The many gates in a same direction form a ray. The many rays of a scan form a sweep. And the many sweeps with different elevations or azimuths form a volume. All of process for radar data follow this structure. In conventional method, radar data retrieval process usually contains several major steps like: raw file reading, data normalization, coordinate transformation and data interpolation. On raw file reading step, radar data is read by different data structure, as it is generated. Several common formats are used like Dorade file, UF file, FORAY netCDF file etc... Through data normalization, the intensity value of each gate can be matched with color scale, then radar data can be presented by images.

The next part is coordinate transformation. Radar emits electromagnetic wave from radar site with a fixed detecting range, the data of radar collected from a spherical coordinate. When we attempt to use the data, it should be transformed into the Cartesian coordinate and interpolated the super-resolution part, it means that we need find or calculate all the value for each grid in Cartesian coordinate (figure 4) based on raw data image. In radar application field, radar image usually has three display description models, PPI (Plan Position Indicator), RHI (Range Height Indicator) and CAPPI (Constant Altitude Plan Position Indicator) [16]. The differences between three models are observing angle and data collection model. Usually the PPI model is most frequently used. In PPI any gate (R, A, E)

from radar can be mapped on a flat surface (x,y) as flowing functions,

$$\begin{aligned} y &= R \times \cos A \times \cos E \\ x &= R_i \times \sin A \times \cos E \\ (i &= 1,2,\ldots,N) \end{aligned} \tag{5}$$

where (x,y) is the location in the Cartesian coordinate, A denotes the azimuth, E denotes the elevation angle. R is the range from radar to number i gate.

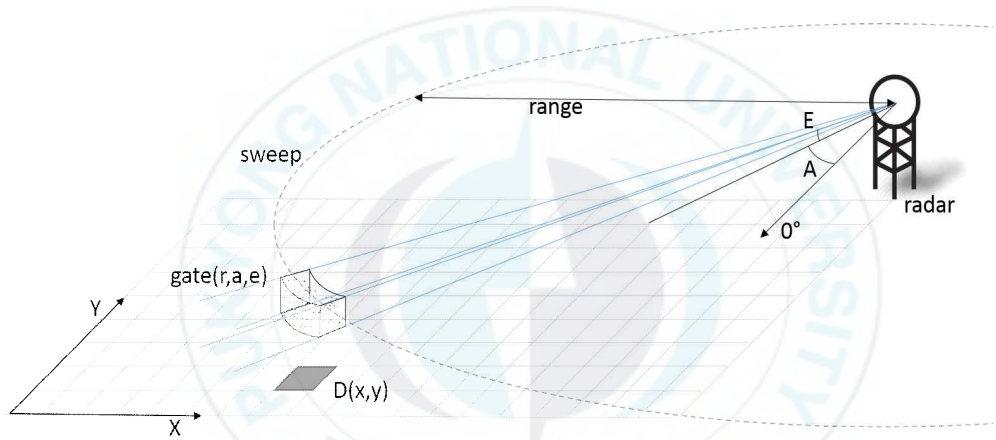


Fig. 4 Coordinate mapping from spheroidal coordinate to Cartesian coordinate.

Because the beams of radar are not continued, data interpolation becomes a necessary step for the super-resolution part after coordinate transformation. As the most computationally expensive step, many algorithms have been proposed, the difference between them are the influence function (estimation function) and the weighting scheme. Nearest neighbor mapping [17] [18] is a simple algorithm, it chooses the value of the nearest point for the non-given points without considering the values

of neighboring at all. But the difference of echoes from adjacent beams become bigger and bigger with radiating out of beams, then radar image will be gridded and discontinuous. Bilinear interpolation [19] is a common algorithm, extended from linear interpolation. It uses four neighbor pixels respectively along two different directions to interpolate. Comparing with nearest neighbor mapping algorithm, the discontinuous of bilinear interpolation can be much improved, but has a little blur on contours. Kriging [20] [21] is an interpolation method originally used in geostatistics. It calculates a point by computing distance weighting of neighbor points which have values and estimate based on Gaussian process. Under suitable assumptions, Kriging method can give smooth and reliable values, but it is computationally expensive and more complicated than the previous two methods. Barnes interpolation [22] [23] describes a method to interpolate data from a set of randomly spaced data in two dimensions using a multi-pass scheme, it is commonly used in meteorology [24]. In all mentioned methods, Barnes has relatively good edge sharpness, smoothness and non-obvious discontinuous, meanwhile it involve amount of calculation. All interpolation methods have two judgement scales: the processing time and smooth reliable data. People attempt to find a fast algorithm meanwhile that looks less gridded. But it is almost impossible to satisfy both of them at same time. In practical applications, because of the timeliness of radar data, people always tend to satisfy the former at first.

2.3 The features of tornadoes in radar image for recognition

Since tornado records, people utilize many features for detecting tornadoes, from single threshold to model simulation. All of these methods need find some features from tornadoes. Like the classic hook echoes [25] [26] from base reflectivity radar image (figure 5), it was first documented by Stout and Huff in 1953, and it is sign to justify issuing a tornado warning by the National Weather Service of the United States[27] [28]. Wang [29] et al. proposed a method for recognizing tornadoes in Doppler radar. It is just based on this feature. However, Forbes [30] found that not all of tornadoes have the hook echo feature in his research, even more than half of them. When tornado appears, it does not always present the classic “hook echo” on radar screen. Figure 6 shows the tornado at 12:46 pm on April 24th in the city of Durant [31]. At that time, the tornado was producing extreme damage which is classified level-four as “EF-scale” [32]. If we only observed "hook echo" from the precipitation of conventional radar, we would never know there was a tornado. The precipitation completely wrapped tornados like figure 6(a), it is not different from its surrounding. But in velocity image (figure 6(b)) can show that winds blow either towards or away from the radar site, because an individual Doppler radar can only measure wind in one dimension. TVS (Tornado Vortex Signature) is the most unique feature to tornadoes. It is used to calculate the POD (Probability Of Detection) of the national network of WSR-88D radar (Weather Surveillance Radar - 1988 Doppler). Thomas [33] proposed what percentage of Doppler radar detected vortices produce tornadoes and what vortex attributeds are most useful in discriminating vortices that are (and are not) associated with a tornado. Besides these, other features are also utilized, like TSS (Tornadic Seismic Signal) [34], people try to detect the seismic energy from ground when tornadoes are in contact with the ground. But it can detect a tornado when it is included in high level of EF-scale and after tornadoes touch down. Since Doppler radar has been applied in weather research, another effective detecting feature "couplets" has always been used to detect tornadoes.

The velocity shows us the wind in precipitation. It is important for people to detect some fine-scale short-time weather phenomena like tornado. Doppler radar can detect cloud is moving toward or away from observer, then to tornado, Doppler radar will find two color from both ends of the color scale is side by side within a very narrow space. That's becausee of tornadoes' fast and strong rotation.

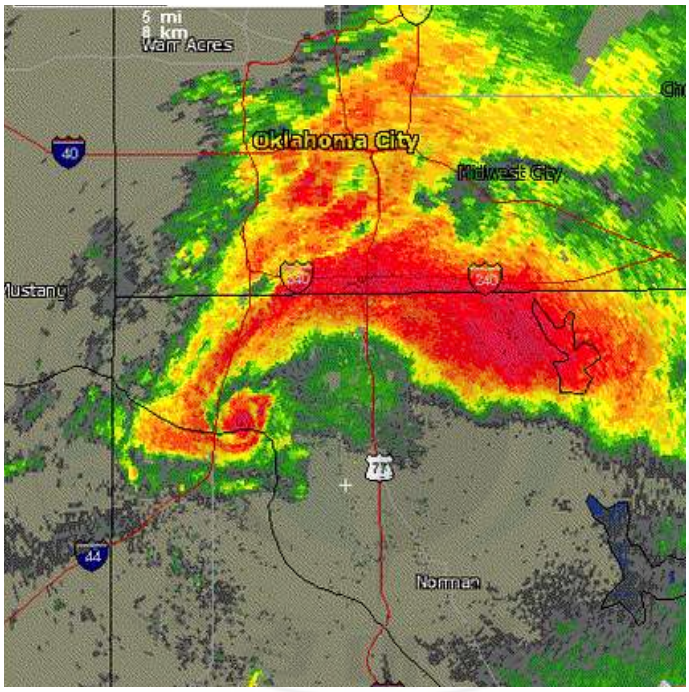
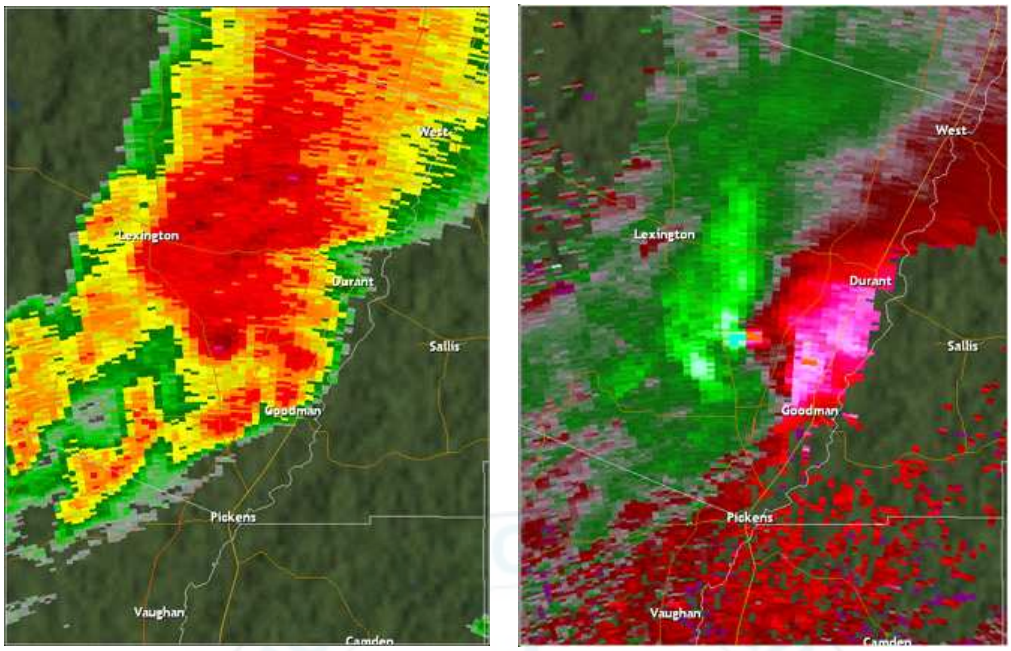


Fig. 5. Hook echo in base reflectivity image.



(a) (b)

Fig. 6 (a) base reflectivity image of Doppler radar, (b) velocity image of Doppler radar.

2.4 Neural networks and back propagation in CNNs

In 1958, David Hubei and Torsten Wiesel found the orientation selective cell in John Hopkins university[35]. Their research found that apart of nerve cell of brain cortex would be active when pupil got some stimulation. And when pupil found the edge of object, if this edge pointed at some direction, the corresponding part of nerve cell would be active. The work of David Hubei and Torsten Wiesel inspired people think deeply. the working process of nerve cell, central nerve and brain, may be a continuous process of iteration and abstraction. According their visual system research, some model have been proposed [36][37]. As previous mentioned, the model extracts and classifies the feature of input image. The learned features are usually very robust to their specific object. Recently CNNs

algorithm even achieved state-of-the-art performances, such as ImageNet Challenge[38] and IJCNN (International Joint Conference on Neural Networks) competition[39][40].

Now the application of CNNs in image processing becomes more and more extensive, including detection, recognition and classification [41][42][43]. CNNs briefly contains two parts: convolutional neural networks structure and back propagation of full connection. Like the conventional neural network, deep learning has similar layered structure (figure 7). The system contains input layer, some hidden layers and output layer. In this structure only adjacent layer has connection, there is no connection in the same layer and cross-layer. Every layer can be seen as a logistic regression model. This kind of layer structure is very similar to our brain. In conventional neural networks training method is not satisfactory. One of the major problems is the gradient will be more and more sparse, when the system gets deeper, the error is smaller and smaller. Another problem is sometime the system is only converge in a local minimum, expecially the start place is far away from the global optimum. For overcoming the problems in neural networks training, deep learning uses back propagation method. The whole networks are trained by iterative algorithm, set initial values randomly, calculate the output. Then algorithm changes the parameters of previous layer according to the difference between last output and labels. This process will continue until convergence (figure 8). The advantage in this process is parameter sharing, sparse interactions and equivariant representations.

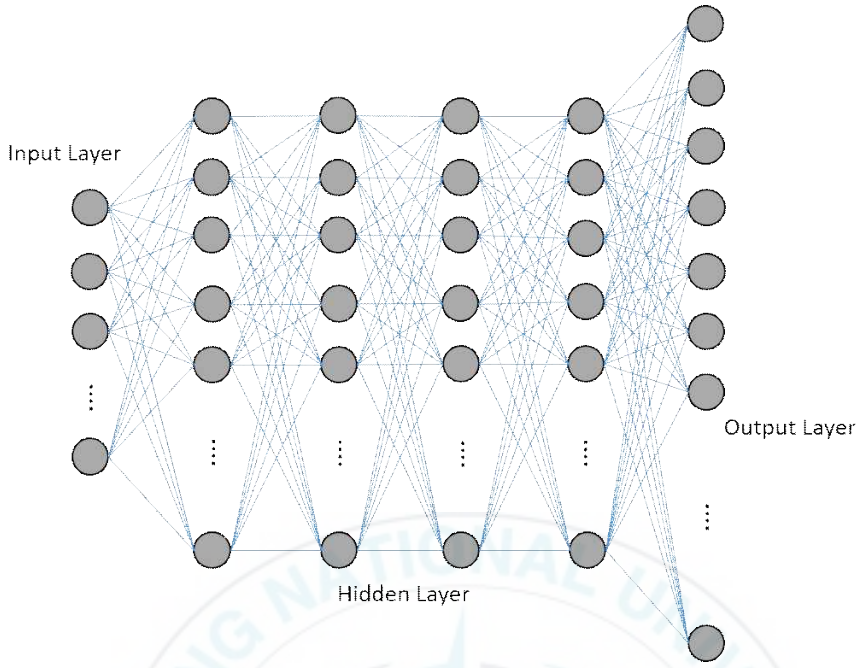


Fig. 7. Layer structure of neural networks.

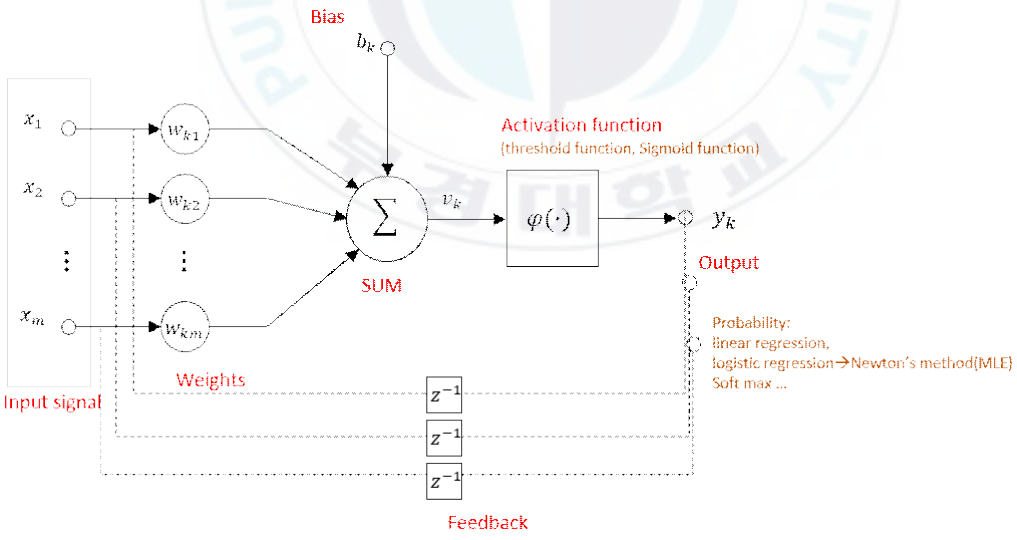


Fig. 8. Back propagation algorithm model.

Ⅲ. Hierarchical contour-line PPI generation based on MRF

In conventional method, radar image generation process mainly contains four parts: raw file reading, data normalization, coordinate transformation and data interpolation. Because of coordinate transformation, not all of grids in the Cartesian coordinate can find corresponding values, more parts need to be estimated. Through analyzing the reference weather radar retrieval system [44], we can find that interpolation process occupies the vast majority of process time in the retrieval system, usually this part can reach 75~95 percentage. Nevertheless the interpolation is just lots of dense repetitive calculation. So we attempt to find a way to avoid or reduce this kind of mass interpolation process, and propose a method, contour-line retrieval of multi-layers based on the MRF. After raw radar file reading, the input data is processed by data normalization and denoising as most of radar data process, then program generate a raw radar image. We use this image as the input, the brief idea is as following figure 9.

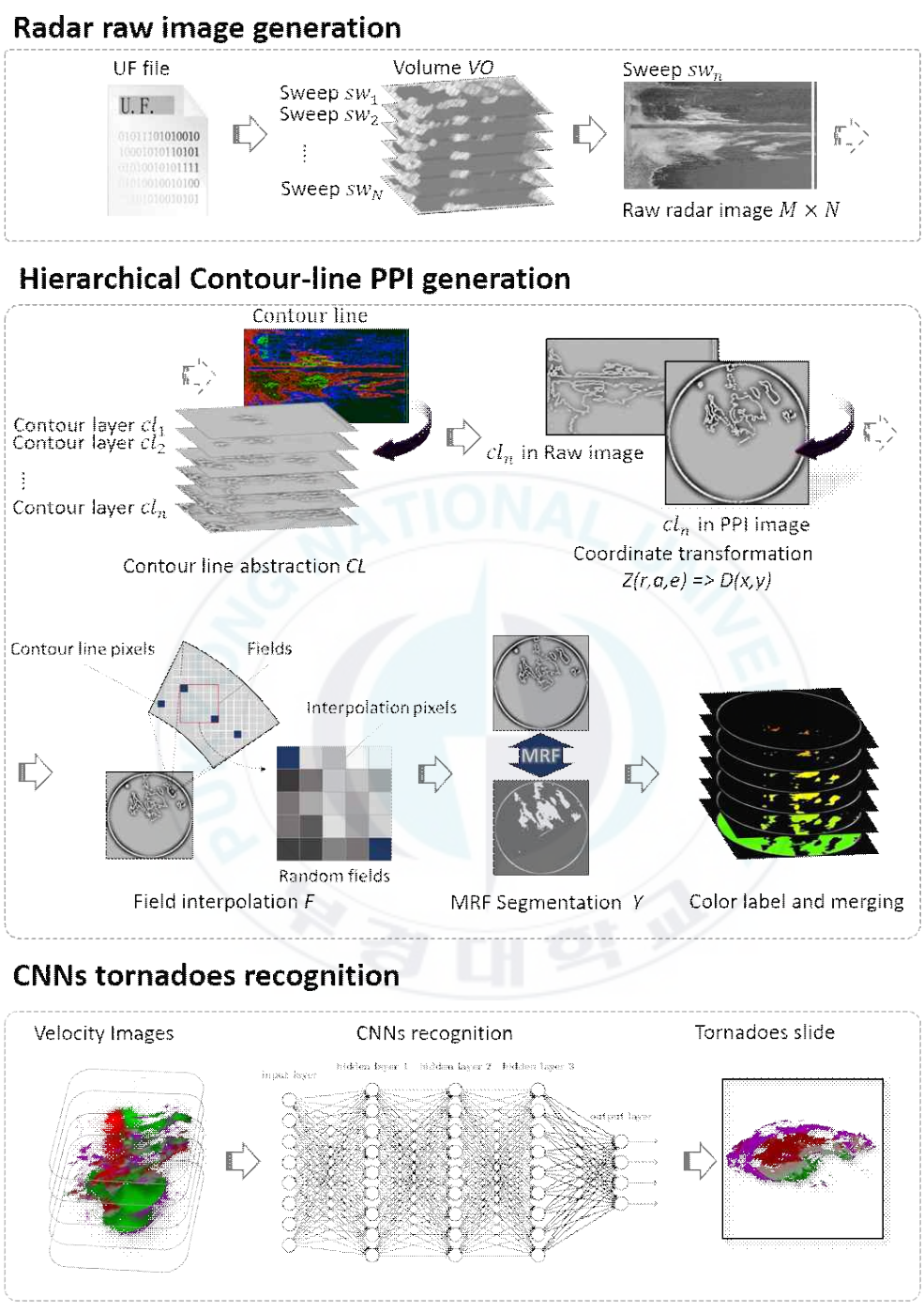


Fig. 9. Flow char of a Tornado Recognition Method by Using CNNs Based on Fast Doppler Radar Image Generation Method.

3.1 UF data Structure

The space which is scanned by radar is called a volume in radar data structure, that radar scan around with a fixed elevation angle is called a sweep SW. When radar send a radar wave, it is called a ray RA. A volume contains a series of sweeps $VO = \{sw_1, sw_2, \ldots, sw_{N_s}\}$, and a sweep contains a series of rays which have same elevation angles $SW = \{ra_1, ra_2, \ldots, ra_{N_r}\}$. All of rays are sampled by the most fundamental unit of radar data, called gate or bin GA and $RA = \{ga_1, ga_2, \ldots, ga_{N_g}\}$. In this paper the raw radar data is generated following the Universal Format (UF) structure. This format is created for Doppler radar data by Barnes [45]. One file contains a complete volume scan of radar. Within a file is a series of stand-alone rays. The data is basically all 16 bit integer words, floating values are expressed by a scale factor. The organizing information of data and other setting information of radar store in header blocks in beginning of each ray recorder. Due to memory-saving purpose, radar scan modes are indicated by number and the field names are only two ASCII characters, like CZ denotes the corrected reflectivity factor $[dBZ]$ and VE denotes the velocity $[m/s]$. A UF raw file of radar can be read following the structure in figure 10, the detail can follow the program manual [46].

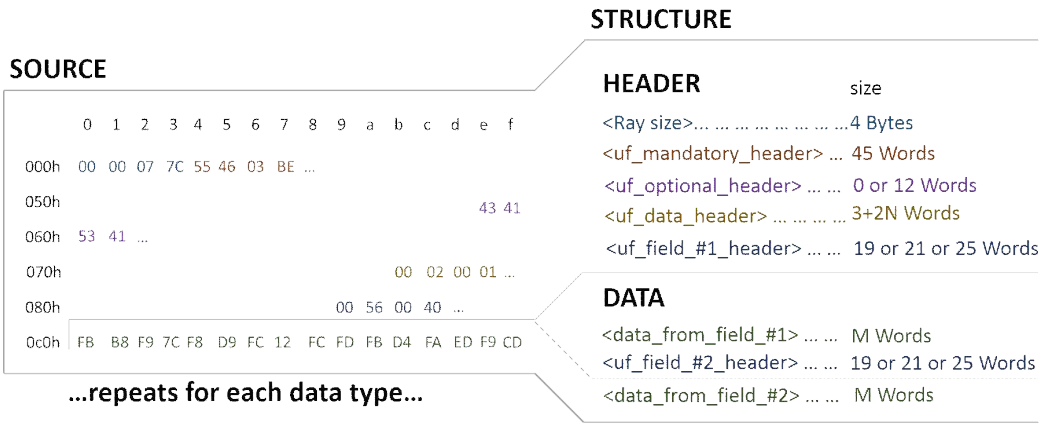


Fig. 10. A brief structure of UF raw file.

3.2 Raw image preprocess and contour line abstraction

3.2.1 Image denoise analysis

Following the UF structure, radar echo intensity can be read gate by gate. They can compose a matrix which the length is equal to the number of gates and the width is equal to the number of rays. After data normalization, a whole sweep of radar scan can be expressed by an image, we call it raw radar image. On this image, every pixel denotes a gate, the radar echo intensity is expressed by image intensity, radial resolution should be equal to the number of pixel in a line and azimuthal resolution should be equal to the number of pixel in column. Therefore this raw image contains all the radar echo intensities and how they distribute. In image processing, usually there are two part contents on an image: contour and color, and most information of image is expressed by contour. So the intensity distribution on PPI radar image can also be described by the

contour line, the amount of computation will be less than the interpolation of large area.

Before contour line abstraction, the raw radar image from radar data reading will be processed by denoising process and thresholding according to reflectivity scale. In denoising process, paper compared three typical denoising algorithms: mean filter, wavelet and Adapted mean filter [47]. Considering keeping the detail of image and removing noise, we selects mean filter for denoising process. Meanwhile the denoising algorithm can smooth the contour line for reducing the calculated amount for post process.

In most applications, radar image uses a kind of 4 bit color scale, so the raw radar image is split up into 16 layers as well. In contour line abstraction, the canny edge operator [48] is used, and as a result, these contour trees can be expressed as sequences [49] [50]. From these sequences we can know the relations between pixels in contour, it is important because part of these pixels will not be continuous any more after coordinate transformation, then the sequences will help us. The result of contour line abstraction will be used in post process, one is that program use the chain sequences of contour to find the neighbor of each pixel for building the fields for MRF segmentation. The other is that program use the area in contour to decide the order in the final layers merging step.

3.2.2 Build random field in PPI image

Instead of all pixels transformation in conventional method, we only transforms the pixels in contour line. The functions are exactly same as the conventional one (function 5). After the transformation, the most part

of pixels in contour line become a series of independent points. Just like opening an umbrella, the raw image can be though like unopened umbrella, and the PPI image is an opened one. The radar site is the center of umbrella, each ray is like a rib of the umbrella. The pixels on "ribs" have exact values from a raw image, but pixels on "panel" is a dataless part, they need to be estimated.

In next step, we generate a complete closed contour line by MRF. Before that we build a series of fields based on adjacent pixels of contour line. For instant, a contour line sequence in raw image $l' = \{i'_1 j'_1, i'_2 j'_2, ..., i'_{N_p} j'_{N_p}\}$ after the coordinate transformation, the pixels locate in new positions in PPI image, then $cl = \{i_1 j_1, i_2 j_2, ..., i_{N_p} j_{N_p}\}$, ith and jth are the location along row and column of image. A rectangle field can be built based on two adjacent pixels, for redundancy, we use $(i_1 - 1, j_1 - 1)$ and $(i_2 + 1, j_2 + 1)$, if $i_1 < i_2, j_1 < j_2$ as diagonal points of the field.

The values in field, we choose bilinear interpolation to calculate. Let R denote the range of radar, N_φ and N_Υ respectively denote the resolution of azimuth and radius, and $Z(\theta, \rho)$ denotes the interesting point. There will be four nearest known points $Z_{i,j}, Z_{i,j+1}, Z_{i+1,j}$ and $Z_{i+1,j+1}$ from two adjacent rays. Then bilinear interpolation estimate of the unknown value is given as:

$$Z = [1 - \varphi, \varphi] \begin{bmatrix} Z_{i+1,j+1} & Z_{i+1,j} \\ Z_{i,j+1} & Z_{i,j} \end{bmatrix} \begin{bmatrix} 1 - \Upsilon \\ \Upsilon \end{bmatrix} \tag{6}$$

where $\varphi = \theta - \frac{2\pi(i-1)}{N_\varphi}$ and $\Upsilon = \rho - \frac{R(j-1)}{N_\Upsilon}$.

3.3 The framework of MRF segmentation

Markov Random Field (MRF) is an image model to describe a field which has random value and Markov property. MRF method is based on MRF image model and Bayesian estimation, it provide a way to build a model for solving the ill-posed problems. In computer vision and image processing the MRF theory has been widely used, since S. Geman and D. Geman [51] proposed the image segmentation algorithm based on MRF. We attempt to use MRF segmentation algorithm to find edge in fields, because the distribution of contour line match the Markov property. In this section, we use MRF - MAP and metropolis algorithm which is talked based on energy and image.

3.3.1 Definition of MRF in an individual radar layer

In each layer of radar image, there are a series of fields $F=\{f_0,f_1,\cdots,f_K\}$ which generate from previous steps. Because the contour line is a closed curve, if a contour line have m pixels then we will get K fields, $K=m-1$. Let us associate a layer image with a stochastic process, based on a set of lattice points S as a field. Then s is a lattice point $s\in S$. Because we attempt to estimate the corresponding classification for each pixel. Suppose is a pixel in a rectangle sub image Y, then for each single $f_K=Y$. y_s is the value of Y at s. Here we use ∂_s to denote the neighboring point of s, and it must be symmetric, which has $r\in\partial_s\Rightarrow s\in\partial_r$ and $s\notin\partial_s$. A clique is a set of points, C. Which are all neighbors of each other. The neighborhood systems and cliques are also depicted as figure 11. Then the MRF can be

expressed as a stochastic process Y on the lattice S with neighborhood system ∂_s, if all $s \in S$:

$$P(y_s|y_r, r \neq s) = P(y_s|y_{\partial_s}) \tag{7}$$

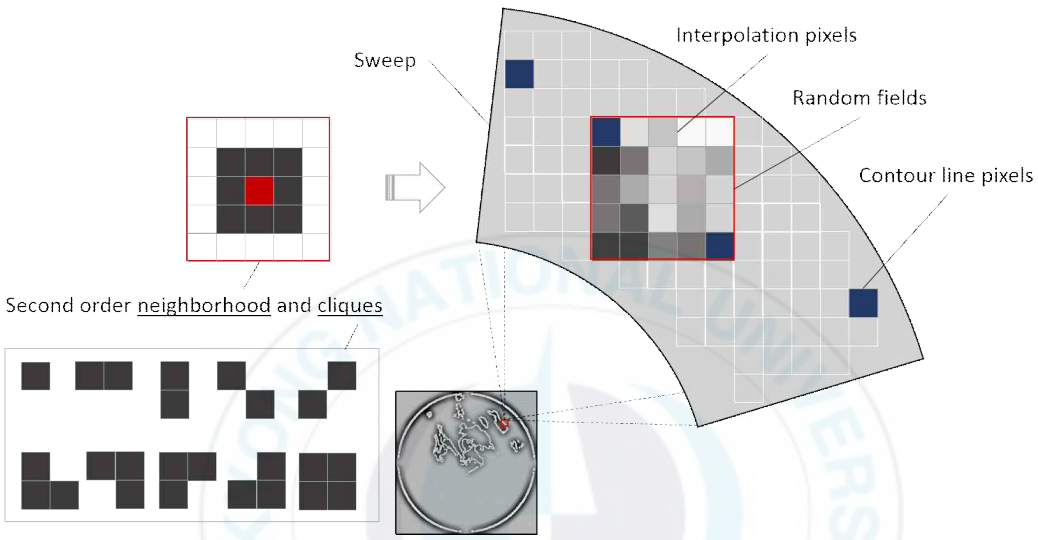


Fig. 11. Definitions of neighborhood and cliques in sweep.

3.3.2 MRF and Gibbs equivalence

Based on previous definition, each field in radar layer can be described by conditional distribution, but this distribution is a "local" property, it is hardly to describe the whole random field. Besag [52] solved it by associating Markov random field with Gibbs distribution. MRF depicts the statistical property of local image, Gibbs distribution can extend it to the global. The association between them is based on a fundamental theorem of random fields, Hammersley-Clifford theorem [53] [54]. It states that a Markov random field can be represented as a Gibbs distribution. Let y_c is value of Y at the points in clique c, $V_c(y_c)$ is the potential function of y_c. Then Gibbs distribution can be written as follows:

$$P(y)=\frac{1}{z}\cdot\exp\left[-\sum_{c\in C}V_c(y_c)\right] \tag{8}$$

Where C is the set of all cliques. Z is partition function. The exponent part also can be expressed as $U(y)=\sum_{c\in C}V_c(y_c)$, It is called energy function, the smaller this value the lower energy. In physics, we know that it is always more stable to be more low energy, and more possibility to realize.

3.3.3 Framework of MRF-MAP

Maximum a posteriori (MAP) is a common optimum criterion, and MAP-MRF is the system combining MRF model with MAP criterion. Let Y denotes the image field, X denotes the labeling field. Then the solution of MAP is the maximum of following function.

$$\hat{x} = \arg max\{P(x|y)\}, x \in X \tag{9}$$

According to the Bayes formula the object function can be written as following:

$$\hat{x} = \arg max\{P(y|x), P(x)\}, x \in X \tag{10}$$

The probability distribution of MRF is equivalent to the distribution of Gibbs, $P(x) = \frac{1}{z} e^{-U(x)}$, then the equation can be written like $P(y|x) = \frac{1}{z} e^{-U(y|x)}$. Then Introduce Gibbs function into the object function, we can get $P(y|x) \cdot P(x) \propto e^{-[U(y|x)+U(x)]}$, then the object function becomes:

$$\hat{x} = \arg max\{U(y|x) + U(x)\}, x \in X \tag{11}$$

For convenient calculation, the Ising model is generally use to describe the potential function. This model is proposed by Ernst Ising in ferromagnetic study [55]. The energy of interacting particles is determined from their spin. The spin of particles has two states "up" and "down", denoted by "+1" and "-1". This is very similar to a binary image. From physic conception, we can know that more reverse spins, the system has more energy, It is more instable. Then the model can be described by the length of edges between reverse spins [56].

$$P(x) = \frac{1}{z} \cdot \exp\left[-\frac{E(x)}{BT}\right] = \frac{1}{z} \cdot \exp\left[\frac{w_{ij} \sum_{i \sim j} x_i x_j}{BT}\right] \tag{12}$$

Where E is the sum of all neighbor pairs of particles, B is the Boltzmann constant, T is temperature, x_i and x_j is the spin of particle, w_{ij} for otherwise. In a finite field, $N \times N$, the vertical boundary edges of particle-pair is $N \times (N-1)$, it is same for the horizontal, and the total edges is $2N \times (N-1)$. Then $\sum_{i \sim j} x_i x_j = 2N \times (N-1) - 2 \cdot dx$, dx is the length of edges of reverse spins. Considering B and T are constants the function 12 can be written as following:

$$P(x) = \frac{1}{z} \cdot \exp\left[\frac{-2w_{ij} \cdot dx}{BT}\right] \tag{13}$$

3.3.4 Metropolis algorithm

MAP estimation reduces to minimizing the posterior energy function [57]. We use Metropolis algorithm for finding a global minima on an image. The Metropolis algorithm is simulated annealing with a fixed temperature [58]. The brief procedure is as following:

1) Start with any state, $x \in C, x = (x_1, x_2, \ldots, x_N)$.
2) Select a pixel from x at random, denoted by x_n.
   Change the sign of its value.
3) The new configuration
   $$x' = (x_1, x_2, \ldots, x_{n-1}, -x_n, x_{n+1}, \ldots, x_N)$$
4) Then accept the new x' depending on the probability as following:

$$\alpha(x'|x) = \min\left\{1, \frac{p(x')}{p(x)}\right\} \tag{14}$$

If $\frac{p(x')}{p(x)} > 1$, then acceptance probability $\alpha(x'|x) = 1$.If $\frac{p(x')}{p(x)} < 1$, the acceptance probability $\alpha(x'|x) = \frac{p(x')}{p(x)}$.

5) Generate uniform random number $u \in U(0,1)$, and accept x' if $u < \alpha(x'|x)$, otherwise, keep x.

Then cycle above steps continually, the system will converge to a stable status from a random status. For each layer of sweep the contour line will stabilize at a certain shape. Because the sizes of all fields are relatively small, the convergence procedure can be completed in a short time.

3.4 Layers completion

In order to get an integrated layer of radar image, we use the flood fill algorithm to fill the area in contour [59] [60]. In radar image, all contour is closed including the boundary of radar screen. Based on that flood fill can be used for radar image. The "seed" can be selected in raw radar image. Different from normal method, we select a seed for background, not for foreground the labeled area. Because the labeled area is several independent disconnected areas in most cases, but the background is not. We need to select many seeds for foreground but only one or few for background. We initially set both foreground and background as the label color, then fill the background with black color or the transparent to complete a layer. The merging order follows the sequence of area in

contour to make sure that there is no area covered by other layer. And the "hole" in layer can be detected by checking the neighbor of starting point in contour sequences.

3.5 Testing environment and raw image generation

In experiment, we use UF raw data file as input to generate the basic reflectivity PPI image of weather radar step by step. The contents of experiment briefly include three parts. In the first section paper will introduce the testing environment including testing data and computer configuration. As preprocess for conventional and proposed methods, the raw radar image will be generated. In the second section paper will introduce the PPI image generation by proposed method and conventional method. At last we will compare two method by computing time and quality performance, meanwhile we will analyze the time distribution in each method.

3.5.1 Testing object and bed

In practical application, radar data retrieval process mainly processes by two computers. One is charge of radar control and signal processing at radar site. The other is charge of data processing or product generation and it can locate everywhere. The former usually uses some powerful server-computers to calculate [61]. But the latter will be not. The simulation of paper uses a normal PC, because the server-computer is already very fast, the improvement on server-computer is not obvious like PC and server-computer is not common like PC. Then in the retrieval

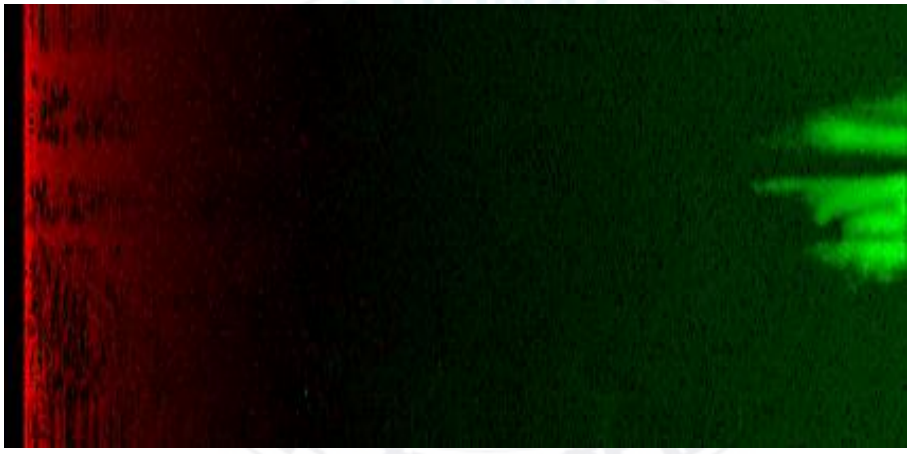
process, a normal computer is used which mainly includes Intel Core2 E7400 CPU and 2GB memory. The simulating software includes VS 2012 and OpenCV based on Window 7 operating system.

The input file is from the CASA project, generated by following UF structure. The raw radar data comes from a radar site located in Rush Springs city, Oklahoma US, and the recoded time is at 7:44 on May 14th 2009. The radar utilizes PPI sweep model with 2 degree elevation. The radar file contains two fields: CZ, corrected reflectivity factor (dBZ) and VE, velocity thresholded on NC (m/s). In CZ field, the number of samples is 426, the number of samples used in volume is 996.

3.5.2 Raw radar image generation

In the raw image generation, we use a pixel to denote a gate in radar file, and match image intensity with the normalized echo intensity of radar. The resolution of raw image is same as the resolution of radar 996*426 (because the elevation is only 2 degree). We follow the UF structure to retrieve the raw radar file as figure 12. In the postprocess the retrieval processes are same for reflectivity and velocity, so in rest part paper only shows reflectivity as example, and utilizes the proposed and conventional methods generate the final PPI radar image.

(a)



(b)

Fig. 12. (a) Base reflectivity raw image, (b) base velocity raw image.

3.6 PPI image generation experiment

3.6.1 PPI image generation with proposed method

Following previous chapter, paper shows the results in the major parts. In the proposed method, the raw image is initially processed by denoiseing algorithm, it's not only for removing some noise, but also for reducing the cost of calculation in post process. We compare the performance with different algorithms like wavelet, mean filter, and adaptive mean filter etc… and the mean filter algorithm is selected [62] [63], because it can keep enough information and has fewer edge points. The denoising performances are as figure 13:

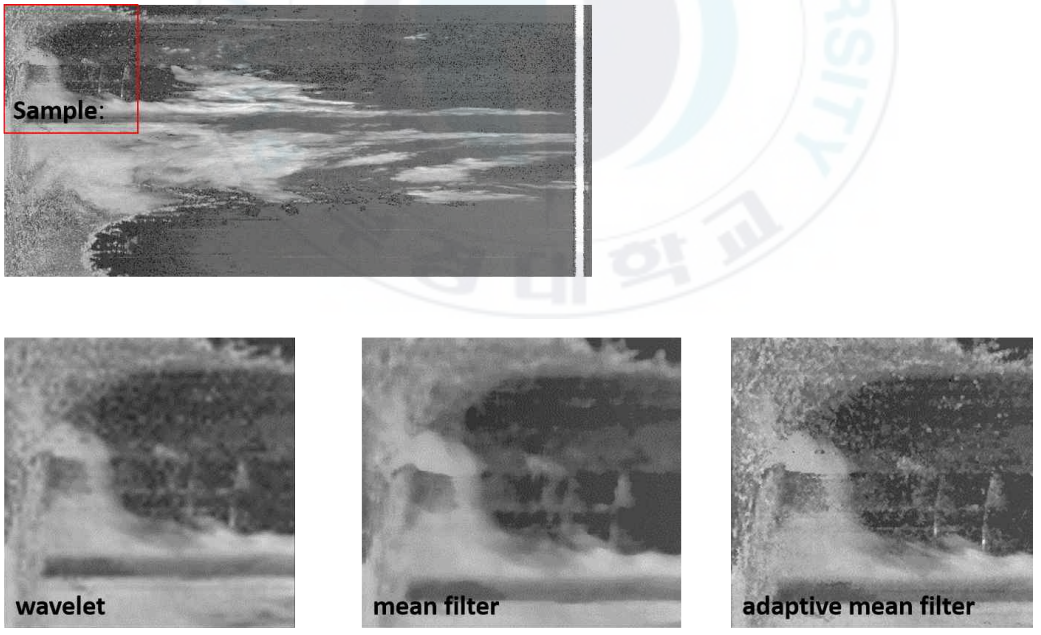


Fig. 13. Performance of different denoising algorithms.

After denoising, we evenly split up the raw image into 16 layers as the reflectivity scale. Usually the reflectivity of radar is a scale roughly between 0 dBZ to 70 dBZ. Because the influence of lower number is very weak, and the higher number rarely occur. But for better performance we use the intensity scale is between -40 dBZ to 50 dBZ, it is the minimum and the maximum value of that day, normalize the scale to of image intensity scale (0~255). Then these are the threshold we used in canny operator. The contour line of the specific intensity threshold can obtain (figure 14). In our case, several layers at either end of the scale has no data, then paper shows some samples of the 16 layers which has reflectivity data.

In next step, the pixels in contour line will be transformed in to PPI image. In PPI image the contour line becomes a series of independent pixels. A whole contour line is stored as a sequence, we can know the neighbors of each pixel. We use two adjacent pixels as diagonal points to build a field, and use bilinear interpolation to calculate a value for filling the fields. Then in PPI image, the pixels of contour line and fields can make a closed contour which can segment the image into "inside" and "outside". After that the flood fill algorithm is used to fill the "inside" part with label color, the result is as figure 15.

Observing about PPI image, the labeled areas are not a whole completed piece, there are some "holes", the lower intensity parts in the area. For avoiding the coverage by unidirectional merging process. We use the color of neighbor pixel of starting point in contour sequences to check the "inside" and "outside" color, make sure they are different. After flood fill process, the closed contour which is composed by fields is not necessary any more. Then MRF algorithm is used to segment these fields into "inside" parts and "outside" parts for generating an accurate edge of the area. After that a completed layer is finished, then we merge these layers

together and get the final PPI radar image following the contour area order in the final step.

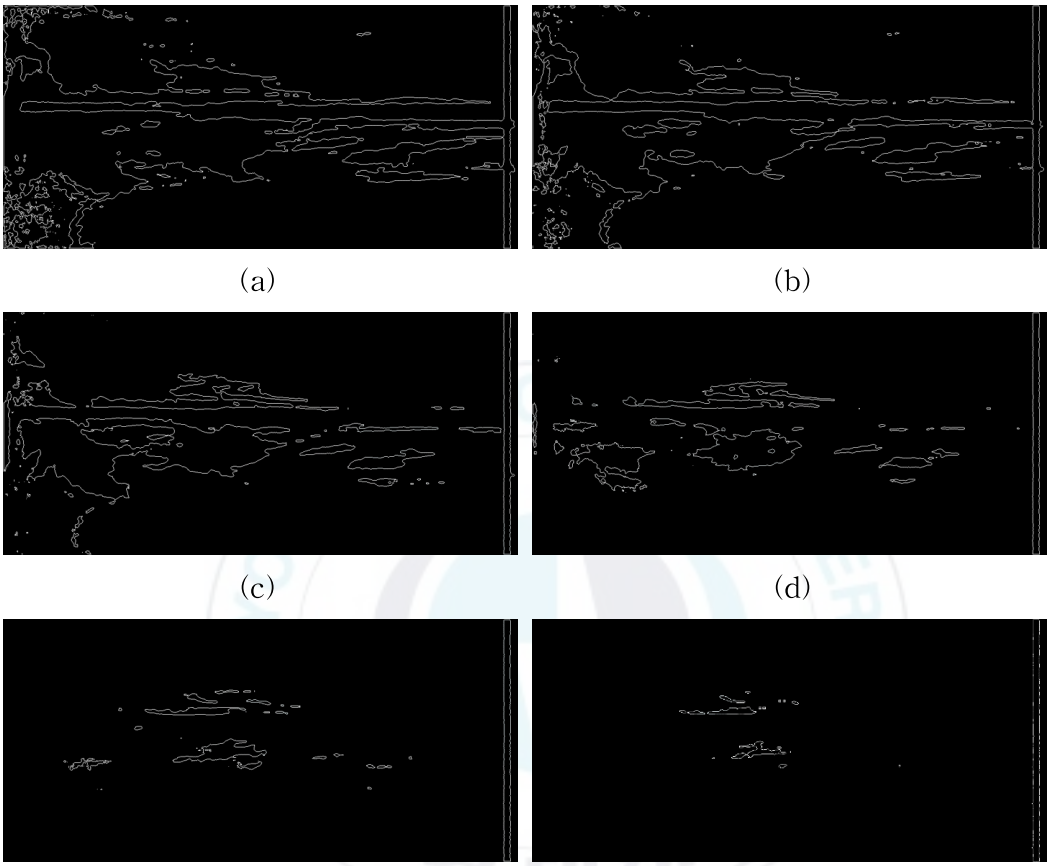
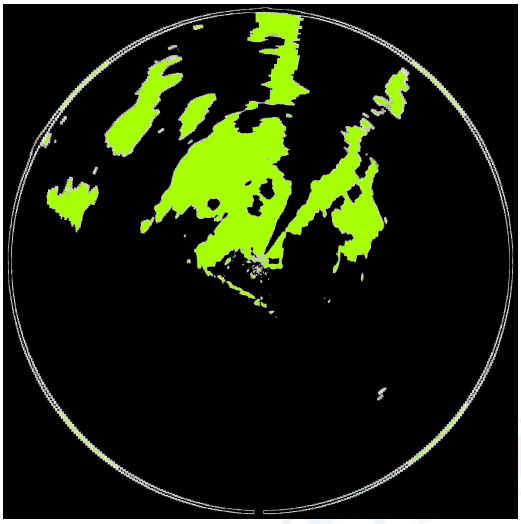


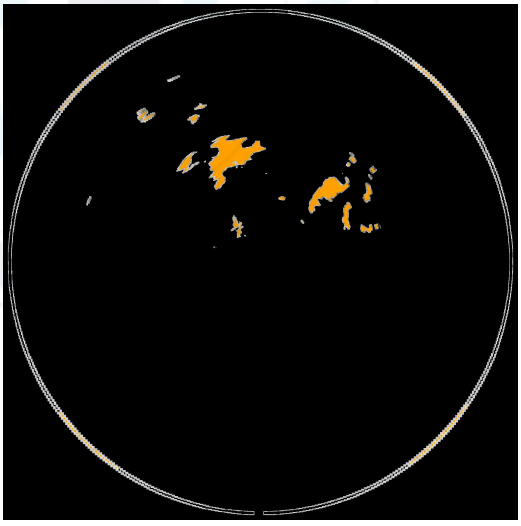
Fig. 14. Contour line in different layers, (a) ~ (f) are 8th ~ 13th layers.



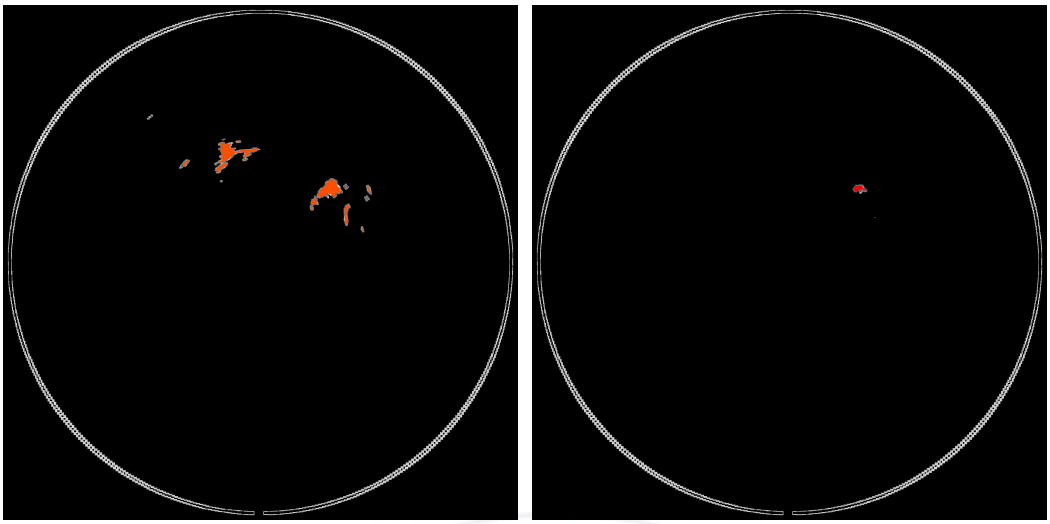
(a)

(b)

(c)



(d)



(e) (f)

Fig. 15. The PPI Image after fields interpolation, (a) ~ (f) are 8th ~ 13th layers.

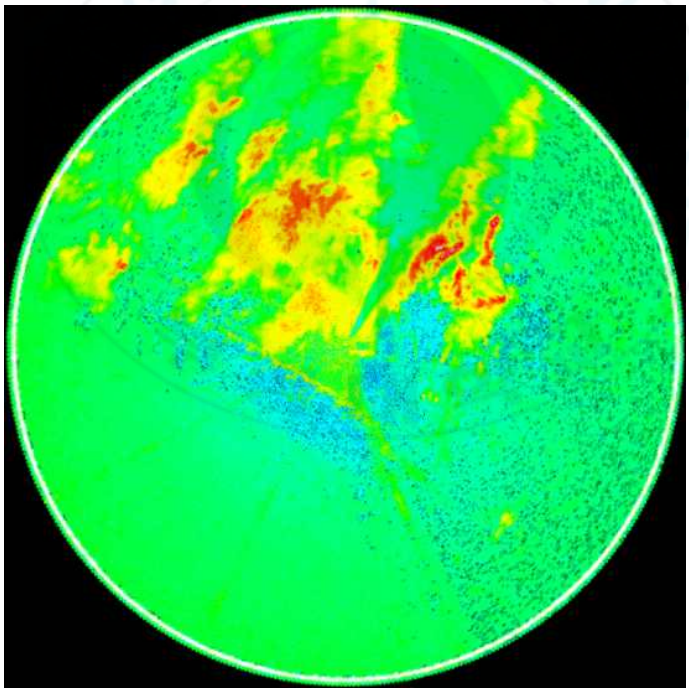
3.6.2 PPI image generation with conventional method

Besides raw file reading step, conventional method mainly contains three steps: data normalization, coordinate transformation and data interpolation. The first two steps have a fixed model as the introduced in related work part. But the interpolation method may be different in different papers. In practical applications, several interpolation methods are commonly used like nearest neighbor mapping, bilinear interpolation, kriging and Barnes interpolation etc… For these methods there are two measurement scales: processing time and continuity performance. In other words, we need a fast and smooth-looking performance algorithm, but it is almost impossible to satisfy both of them at same time. Then we select a fast and frequently used algorithm bilinear interpolation, even it has some insufficiencies [64], because the main purpose of this paper is to improve

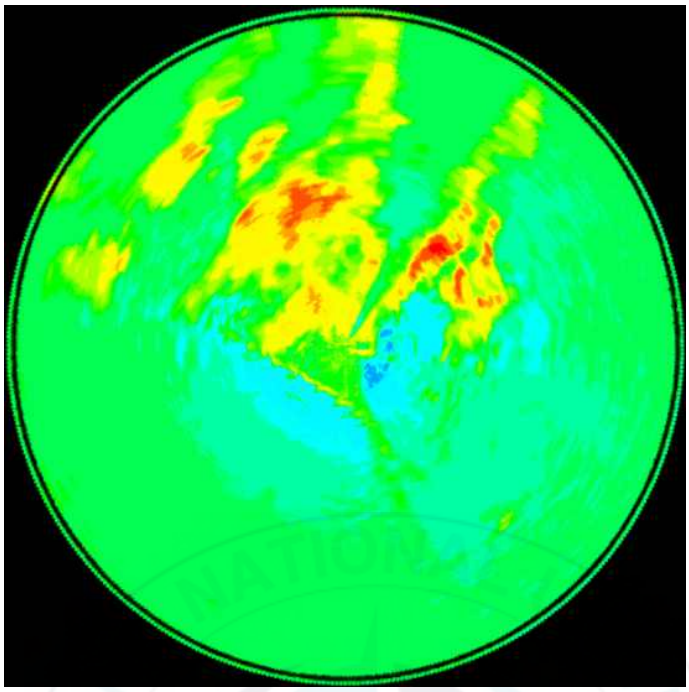
processing time. The conventional method with bilinear method is as R. Kvasov proposed [65].

3.6.3 Quality evaluation and computing time

This section provides an analysis of proposed method, and compared with conventional one. We respectively simulate the conventional method and proposed method, and analyze the performance. Because of the mean filter, the radar image generated by proposed method looks less gridded and more smooth than the conventional, but meanwhile the former losses some details (figure 16).



(a)



(b)

Fig. 16. Base reflectivity PPI image (a) by conventional method, (b) by proposed method.

About the processing time we compared 10 radar samples, and the result of comparison is as figure 17. From figure we can find that the proposed method has much improved the processing time through avoiding the mass dense interpolation process in conventional process [66]-[69]. Through observing the proposed method step by step, two major computationally expensive steps are field interpolation and MRF segmentation. Both of them occupy 97 percentage of the whole processing time (figure 18). One further step, through analyzing the processing time of each layer, we can find that the determining factor of processing time of two major steps is the number of fields. And the number of fields is equal to the number of pixels in contour line. Figure 19 reflects the relation between the number

of pixels in contour line and the processing time of two major processes. Then back to denoising step, we use the mean filter, because after this filter we can get a relatively simple contour line. That means less calculation amount in post-process[70]-[73].

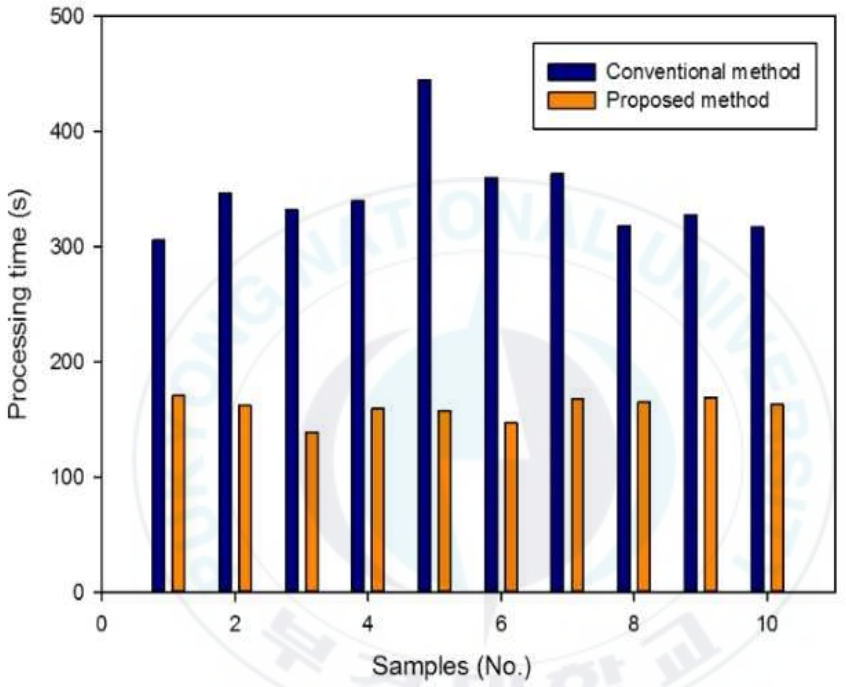


Fig. 17. Comparison of processing time between conventional method and proposed method.

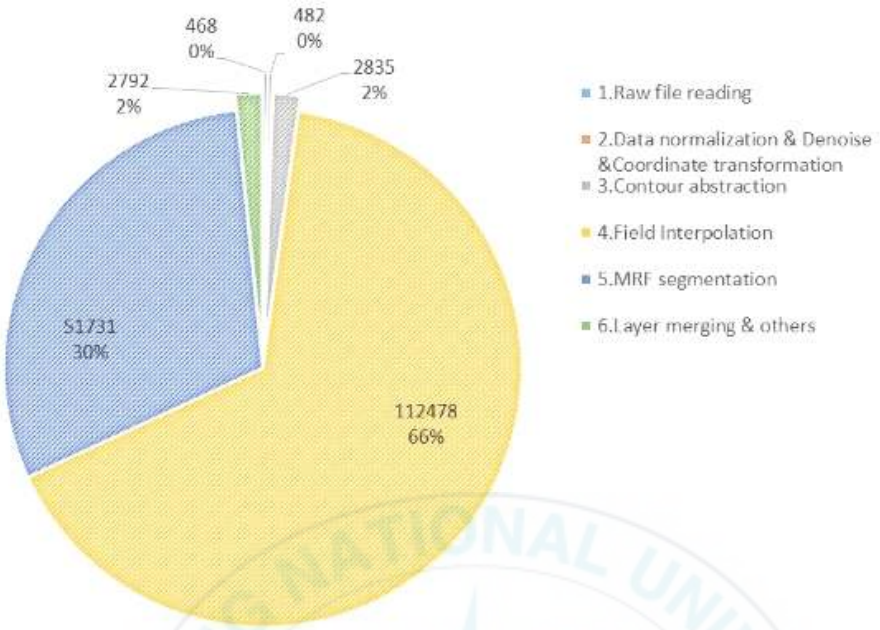


Fig. 18. Processing time distribution of proposed method on average (ms).

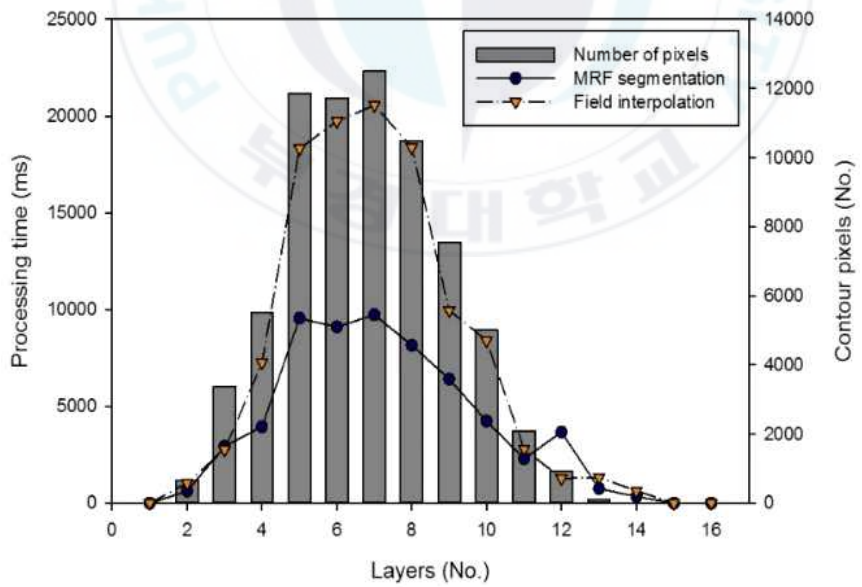


Fig. 19. The influence of the number of contour pixels to the processing time of MRF segmentation and the field interpolation.

Ⅳ.Tornadoes features and regression model analysis based on CNNs

In a typical CNNs, the initial parts are the alternate of convolution and subsampling. At the last some layers, the layers are all one dimensional network of full connection. Then we complete the mission which transforms all two dimensional features into one dimensional network. Next we talk about BP (Back propagation) algorithm.

4.1 Basic regression model

CNNs is supervised regression model among a deep learning algorithms. The process makes prediction according to input features. Supervised algorithm is that finds a relatively accurate prediction with giving training data set. This kind of algorithms can briefly be expressed as figure 20. The linear regression is the most basic and core algorithm. If we supposed that m is the size of training set, x is the feature of input, y is the feature of output. $x^{(i)}$ is the feature vector of sample i of training set, $x_j^{(i)}$ is the feature j of sample i. For single variate the prediction can be expressed as: $h_\theta(x) = \theta_0 + \theta_1 x$. θ is the parameters to be solved.

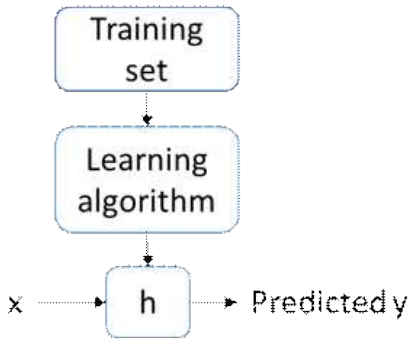


Fig. 20. Brief Learning process model.

In supervised learning process, data set usually contain two parts: training data set and testing data set, for getting a relatively better prediction result. Here the cost function should be defined:

$$J(\theta_0,\theta_1)= 1/2m\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)^2 \tag{15}$$

when we solve the parameter θ_0 and θ_1. our target is to minimize the cost function $\min J(\theta_0,\theta_1)$.

The gradient descent method can be used to solve the parameter. The core idea is that initialize θ_0 and θ_1, continuedly modify and tune θ_0 and θ_1 to get the minimum $J(\theta_0,\theta_1)$. The algorithm can be expressed as: iteration $\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1)$, until it is convergence. But most of time we explain cost function from probability angle. the partial derivative on the cost function can be expressed as:

$$\begin{aligned}\frac{\partial}{\partial\theta_j}J(\theta) &= \frac{\partial}{\partial\theta_j}\frac{1}{2}(h_\theta(x)-y)^2 \\ &= 2 \bullet \frac{1}{2}(h_\theta(x)-y) \bullet \frac{\partial}{\partial\theta_j}(h_\theta(x)-y) \\ &= (h_\theta(x)-y) \bullet \frac{\partial}{\partial\theta_j}(h_\theta(x)-y) \\ &= (h_\theta(x)-y) \bullet \frac{\partial}{\partial\theta_j}(\sum_{i=0}^{n}\theta_i x_i - y) \\ &= (h_\theta(x)-y)x_j\end{aligned} \tag{16}$$

Then for a single training example, this gives the update rule, the Least Mean Squares (LMS) (figure 21).

$$\theta_j := \theta_j - \alpha\left(y^{(i)} - h_\theta(x^{(i)})\right)x_j^{(i)} \tag{17}$$

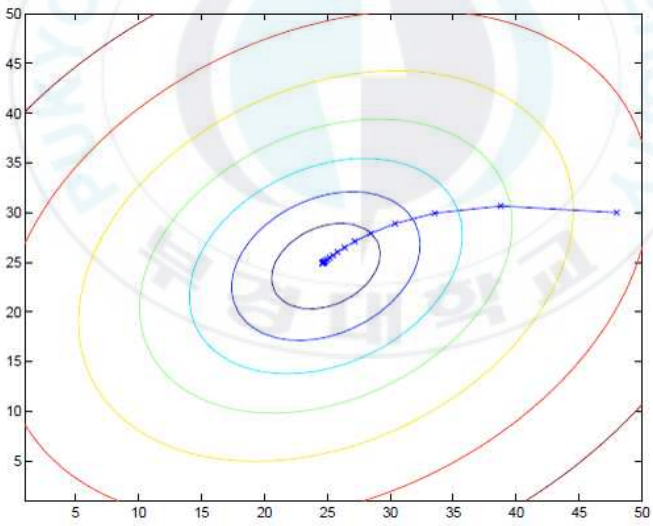


Fig. 21. Linear regression model.

4.2 The algorithms related with CNNs regression model

In practical application basic model is rarely used, because of the amount of repeating data calculation. Then we talk about the major algorithms in CNNs from two area: vision layer and loss layer.

4.2.1 Convolutional layer

The convolutional layer in CNNs is composed by convolutional unit (figure 22). The parameter or weight of every convolutional unit will be optimized by back propagation. The aim of convolutional process is to abstract feature, the shallow layer maybe just abstract some basic features, like edge, line and angle. Deeper layers can abstract more complicated features based on features iteration of shallow layers.

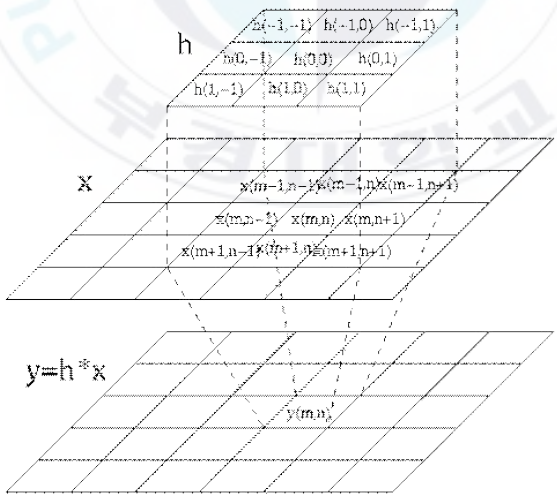


Fig. 22. Convolutional process in convolution layer.

4.2.2 Pooling layer

Pooling layer usually follows convolutional layer, the aim of pooling layer is to reduce the calculated amount, or we say it compress the convolutional layer. Max pooling is frequently-used. For instant one neure matches a 2 by 2 are in convolutional layer, then every neure value match the maximum output from convolutional layer (figure 23). meanwhile it provides rotary deformation.

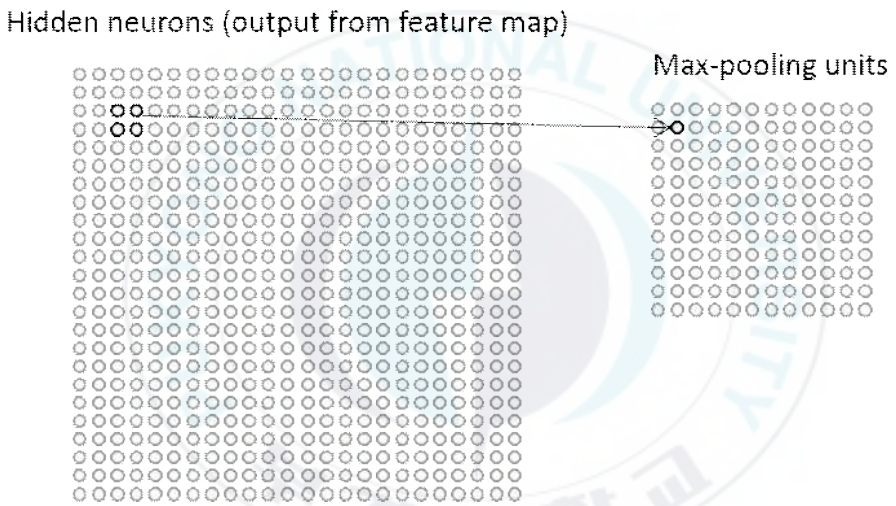


Fig. 23. Max Pooling with 2*2 kernel.

4.2.3 Loss layer - softmax

Loss drives learning by comparing an output to a target and assigning cost to minimize. The loss itself is computed by the forward pass and the gradient to the loss is computed by the backward pass. The softmax loss layer computes the multinomial logistic loss of the softmax of its inputs. It's conceptually identical to a softmax layer followed by a multinomial

logistic loss layer, but provides a more numerically stable gradient.

4.3 Feed forward pass

If we have N samples, belong to C kinds. Then the square error cost function can be written as following.

$$E^N = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{C}\left(t_k^n - y_k^n\right)^2 \tag{18}$$

Where t_k^n is the label of sample n in dimension k. y_k^n is the k output of sample n. For multi-classes question, the output is usually a form "one of c". It means that only the output of corresponding class is positive, the other classes are zero or negative. This depends on the activation function.

Because the error of all training set is the sum of error of each sample, we initially talk about the BP of one sample. To the sample n , the error can be written as following:

$$E^n = \frac{1}{2}\sum_{k=1}^{c}\left(t_k^n - y_k^n\right)^2 = \frac{1}{2}\|t^n - y^n\|^2 \tag{19}$$

In conventional neural networks, we need to calculate cost function E to every partial derivative of each weight according on BP. Use l to denote layers, then the output can be written as following:

$$x^l = f(u^l), with\ u^l = W^l x^{l-1} + b^l \tag{20}$$

There are several activation functions, like sigmoid, TanH (Hyperbolic

Tangent) and ReLU (Rectified-Linear). like sigmoid, it will normalize output into [0,1]. So the last average of outputs always tends to zero.

4.4 Back propagation pass

The error from back propagation pass can be seen as the sensitivities of each neure. the definition is as following:

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial u}\frac{\partial u}{\partial b} = \delta \tag{21}$$

Because $\frac{\partial u}{\partial b} = 1$, then $\frac{\partial E}{\partial b} = \frac{\partial E}{\partial u} = \delta$. It means that the sensitivities of bias $\frac{\partial E}{\partial b} = \delta$ is equal to the derivative $\frac{\partial E}{\partial u}$ of error E of all input u. This is the main idea which let the error of high level back propagate to button level. the back propagation uses function as following:

$$\delta^l = (W^{l+1})^T \delta^{l+1} \circ f'(u^l) \tag{22}$$

The "$\circ$" means each element multiply all other elements. the sensitivities of output level is different:

$$\delta^l = f'(u^L) \circ (y^n - t^n) \tag{23}$$

In the last, use δ to renew the weights of all neure. For a specific neure, get its input, then use the δ to calculate, and express by vector. To the level l, that use error to derived of each weight is the product of input and sensitivities. Then the partial derivative multiply a negative learning

rate is renewed weight of neure.

$$\frac{\partial E}{\partial W^l} = x^{l-1}(\delta^l)^T \tag{24}$$

$$\triangle W^l = -\eta \frac{\partial E}{\partial W^l} \tag{25}$$

Where W_{ij} is weight, η_{ij} is learning rate.

4.5 Convolutional neural networks

In a convolutional layer, Feature maps are convoluted by a convolutional kernel, and pass activation function. Then the output feature map can be gotten. Each output map can be input maps value of many convolutions.

$$x_j^l = f\left(\sum_{i \in M_j} x_i^{l-1} * k^{l_{ij}} + b^{l_j}\right) \tag{26}$$

Where M_j is the set of input maps. Every output map will be given a extra bias b, but to a specific ouput map, the input maps of convolutional kernel is different. It means that if output feature map j and output feature map k, both of them are gotten by input map i, the convolutional kernels are different.

We assume there is a sub sampling layer $l+1$ for each convolutional layer l. From BP, we know that the sensitivities δ of each neure need to be calculate before we calculate the renewed weight of each neure in layer l. For getting the sensitivities, we need initially get the sum of sensitivities in next layer, then multiply corresponding weights W, multiply

derivative of activation function f of input u in current layer. Then the sensitivities δ of current layer can be gotten.

However, because of sub sampling, the sensitivities δ of a neure of sub sampling matches a pixel of output map of convolutional layer. So each neure of map of layer l only connects one neure in next layer $l+1$.

For effective calculate sensitivities, we need to up sample the sensitivities of this down sample layer. So it can make this map of sensitivities equals to the map of convolutional layer. Then use the partial derivative of activation value of map to multiply the sensitivities map of layer sub sampling layer.

The weights in sub sampling layer use a same constant value β. So we can get sensitivities δ in layer l just by multiplying the result from last step by β. We can repeat the same calculative process to each feature map in convolutional layers. But this needs corresponding map from sub sampling.

$$\delta_j^l = \beta_j^{l+1}\left(f'\left(u^{l_j}\right) \circ up\left(\delta_j^{l+1}\right)\right) \tag{27}$$

"up" means up sampling. if the sampling factor is n. If copy n times along vertical and horizontal direction of each pixel, then can restore the size. This function can also realize by Kronecker product.

$$up(x) \equiv x \otimes 1_{n \times n} \tag{28}$$

To a specific map, we can complete the sensitivities calculation. Then we can use summation of all neure of map in layer l to calculate the gradient of bias:

$$\frac{\partial E}{\partial k^{l_{ij}}} = \sum_{u,v} \left(\delta^{l_j}\right)_{uv} \left(P_i^{l-1}\right)_{uv} \tag{29}$$

Where $\left(P_i^{l-1}\right)_{uv}$ is the patch when X_i^{l-1} multiply by k_{ij}^l. The output convolutional map on location (u,v) is the product of convolutional kernel k_{ij}^l and the patch from last layer on location (u,v).

Here the most difficult part is sensitivities map calculation. We must find that the patch in input map matches which pixel of output map. Then the function A can be used, the sensitivities will be back propagation, in addition, it needs multiply the weight between input patch and output pixel. Actually this weight is the convolutional weight $\delta^{l_j} = f'\left(u^{l_j}\right) \circ conv2\left(\delta^{l+1_j}, rot180\left(k^{l+1_j}\right), 'full'\right)$.

Until here, we can calculate the gradient of b and β. Firstly bias b is like previous convolutional layer, just sum all elements of sensitivities map:

$$\frac{\partial E}{\partial b_j} = \sum_{u,v} \left(\delta^{l_j}\right)_{uv} \tag{30}$$

Because the β is related to the calculation of down sampling map in front propagation. We need keep these maps for back propagation. Then we assume:

$$d^{l_j} = down\left(x_j^{l-1}\right) \tag{31}$$

then the gradient of β can be calculated as:

$$\frac{\partial E}{\partial \beta_j} = \sum_{u,v} \left(\delta^{l_j} \circ d^{l_j}\right)_{uv} \tag{32}$$

4.6 Learning combination of feature maps

In most of cases, we initially input many maps into convolution, then sum of the convolutional values to get a output map. In reference papers, generally people select maps to get a output map. Here we let CNNs to combine these maps in training process. It means let the networks to select which maps combination is the best. We use α_{ij} to denote the weight or contribution of input map i of output map j. Then the output map j can be written as:

$$x^{l_j} = f\left(\sum_{i=1}^{N_c} \alpha_{ij}\left(x^{l-1_i} * k^{l_i}\right) + b^{l_j}\right) \tag{33}$$

It is need to satisfy the constraint: $\sum_i \alpha_{ij} = 1, \text{and } 0 \le \alpha_{ij} \le 1$. these constraints of α_{ij} can be expressed as softmax function of a set of unconstraint hidden weights c_{ij}:

$$\alpha_{ij} = \frac{\exp(c_{ij})}{\sum_k \exp(c_{kj})} \tag{34}$$

To a fixed j, every weight c_{ij} is independent to other weights. Then we can remove index j, only consider a renew of map, the others are same, only index j is different. the derivative express of softmax:

$$\frac{\partial \alpha_k}{\partial c_i} = \delta_{ki}\alpha_i - \alpha_i\alpha_k \tag{35}$$

The δ is Kronecker, the derivation of error of α_i in layer l:

$$\frac{\partial E}{\partial \alpha_i} = \frac{\partial E}{\partial u^l}\frac{\partial u^l}{\partial \alpha_i} = \sum_{u,v}\left(\delta^l \circ \left(x^{l-1_i} * k^{l_i}\right)\right)_{uv} \tag{36}$$

Finally we can get the partial derivative of cost function of weight c_i by the chain rule.

$$\frac{\partial E}{\partial c_i} = \sum_k \frac{\partial E}{\partial \alpha_k}\frac{\partial \alpha_k}{\partial c_i} = \alpha_i\left(\frac{\partial E}{\partial \alpha_i} - \sum_k \frac{\partial E}{\partial \alpha_k}\alpha_k\right) \tag{37}$$

4.7 Enforcing sparse combination

For constrain that α_i is sparse, means make a output map to connect some maps, not all of them. We add constraint term $\Omega(\alpha)$ into the whole cost function. To a specific sample, rewrite the cost function as:

$$\widetilde{E}^n = E^n + \lambda \sum_{i,j} \left|(\alpha)_{ij}\right| \tag{38}$$

Then find the contribution to the derivation of weight c_i. the derivation is:

$$\frac{\partial \Omega}{\partial \alpha_i} = \lambda sign(\alpha_i) \tag{39}$$

Then use the chain rule, the derivation of c_i is:

$$\frac{\partial \Omega}{\partial c_i} = \sum_k \frac{\partial \Omega}{\partial \alpha_k} \frac{\partial \alpha_k}{\partial c_i} = \lambda \left(|\alpha_i| - \alpha_i \sum_k |\alpha_k| \right) \tag{40}$$

So, the gradient of weight c_i is:

$$\frac{\partial \widetilde{E}^n}{\partial c_i} = \frac{\partial E^n}{\partial c_i} + \frac{\partial \Omega}{\partial c_i} \tag{41}$$

4.8 Data sets and testing bed

In CNNs process, we use 800 images to train the model, and analysis the loss and accuracy. We talk about some experience in CNNs traning. In tornado and other radar images collection, we use the dataset from NSSL (The National Severe Storms Laboratory) and CSWR (Center For Severe Weather Research). For regular weather radar image is easily to find, But tornadoes are not. So in our training dataset there are 400 images of velocity of tornadoes and 400 images of regular weather (figure 24). The testing dataset are 200 images of tornadoes and 200 images of the regular. In testing process, we only used CPU. The configuration of computer is Intel i5-2500 CPU, 8 GB memory and 64 bit Windows 7 operating system.

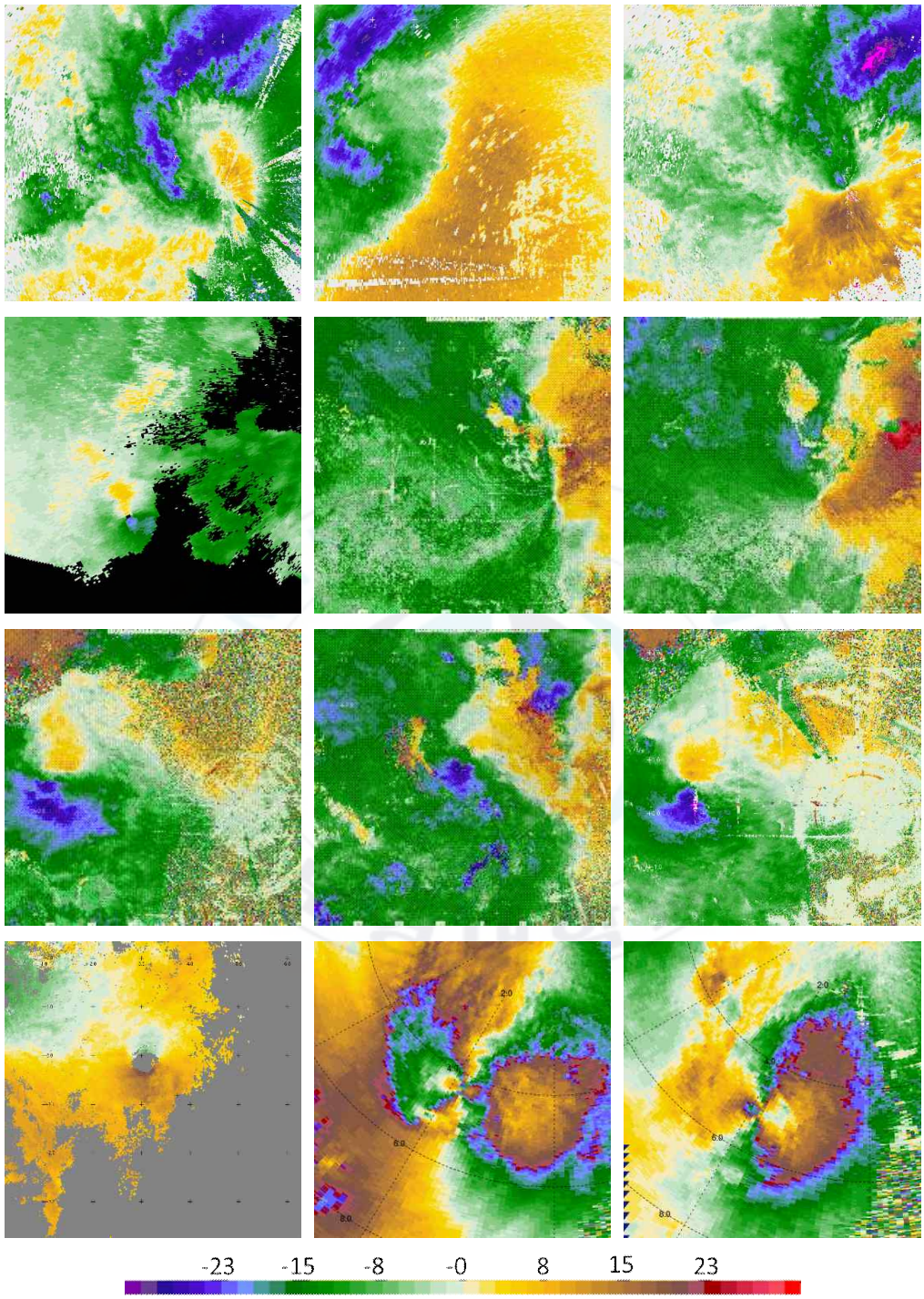


Fig. 24. Samples of CNNs training (tornadoes and regular velocity images).

4.9 CNNs recognition experiment

4.9.1 Testing model

Network design depends on input image size, kernel size of each layer and processing objects, for convenience, our testing model is modified from the ImageNet. The transfer learning the new networks from the imageNet pretrain requires much less data than training from scratch. The whole structure contains eight layers. Input images are conditioned to a specific size 256*256. The image data and label are packaged in two LevelDB files one for training another for testing. The whole model contains eight layers (figure 25), Both of training model and testing model have similar structures (Appendix B). And the data size and features distribution are as table 3.

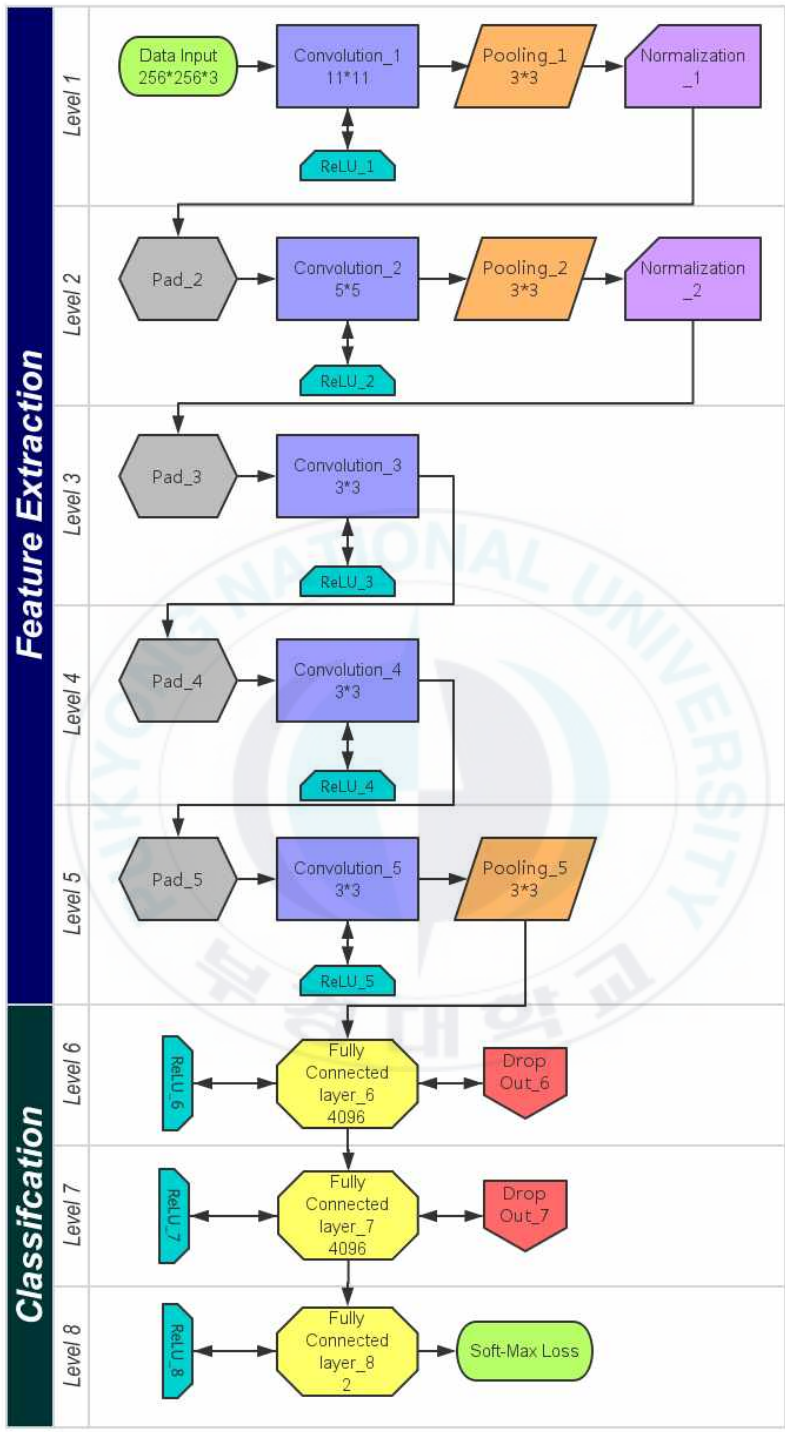


Fig. 25. Layer structure in CNNs.

Table. 3. Data and features in each layer.

Layer	Dimension
Training Data	256*256*3
Training Data (Data Aug)	227*227*3
Convolution 11-96	55*55*96
Max pooling 1	27*27*96
LRN	27*27*96
2. Convolution 5-256	27*27*256
Max pooling 2	13*13*256
LRN	13*13*256
3. Convolution 3-384	13*13*384
4. Convolution 3-384	13*13*384
5. Convolution 3-256	13*13*256
Max pooling 5	6*6*256
6. FC-mine 6	1*1*4096
Dropout	1*1*4096
7. FC-mine 7	1*1*4096
Dropout	1*1*4096
8. FC-mine 8	1*1*2
Softmax	

4.9.2 Training and Testing

In the training, we use "step" learning rates, and the initial value is 5e-5. and the solver parameters are as following:

test_iter: 50
test_interval: 100
base_lr: 5e-005
display: 100
max_iter: 4000
lr_policy: "step"
gamma: 0.1
momentum: 0.9

weight_decay: 0.0005
stepsize: 500
snapshot: 1000
snapshot_prefix: "Tornado_imagenet_train"
solver_mode: CPU

Before the training we need calculate the beverage value and normalization. We totally ran 4000 iteration by CPU. Because using a small dataset and relatively bigger learning ratio, the loss reducing (figure 26) and accuracy increasing (figure 27) are very fast at the beginning 800 iterations. After that, the performance is continually improved with iteration time, but the improvement will be not outstanding like the beginning. If we want to reach the state-of-the-art, more data and more time training are necessary. Meanwhile it needs fine-tuning.

In our tuning experience, the dimension and sparse of data influence the result. Usually ignore the training cost, the more data the better. In training process, some parameters and methods which we used have more influence than the algorithm itself, like pooling and convolution. To image recognition, usually more features are helpful, meanwhile we need sufficient data for the training, otherwise the data will become relatively sparse to specific feature.

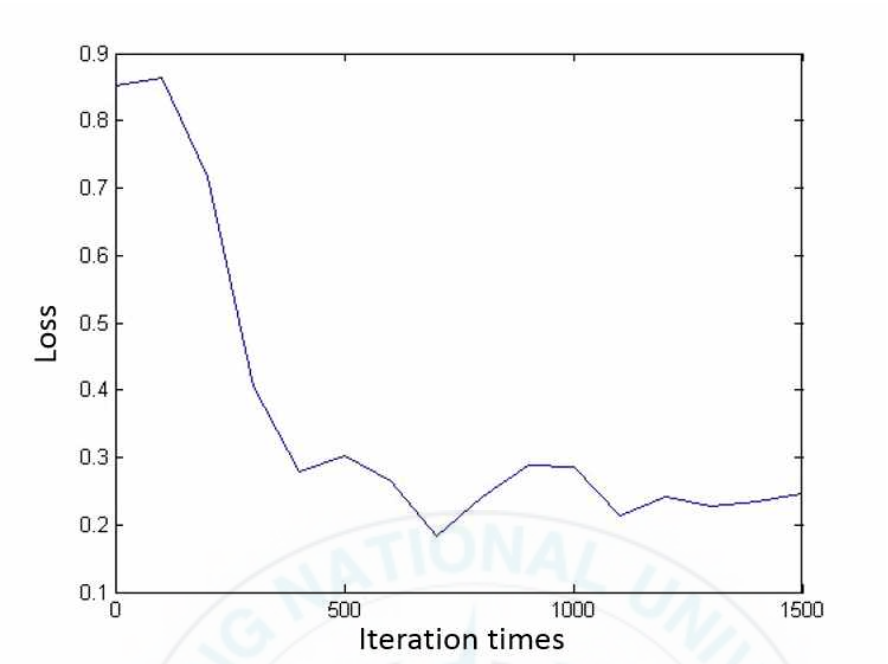


Fig. 26. Loss reduced in 1500 iterations.

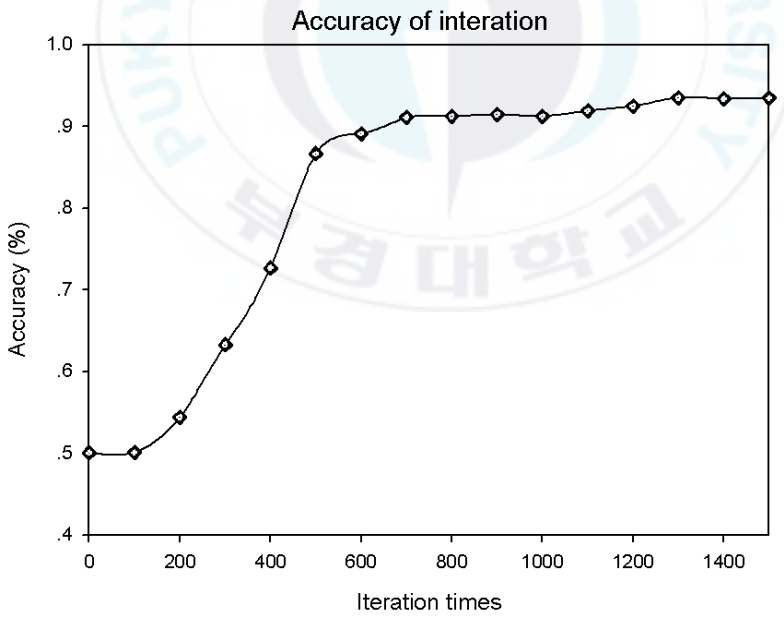


Fig. 27. Accuracy in training.

4.9.3 Reference methods and features

People have tried to predict tornadoes for a long time. But the observations are limited by performance of radar. Every proposed algorithm is based on their observation equipment at that time. There are simple threshold recognitions, pattern recognitions of the base reflectivity of conventional radar and base velocity recognitions of long range Doppler radar [74]-[78]. Until Doppler radar application, people can observe inside of tornadoes. Based the velocity image of radar the tornadoes can be visualized observed [79]-[81]. We know that the Doppler radar can only detect the speed of radial direction. If we want to know the wind direction (figure 28) in tornadoes at least two Doppler radar are needed. But because tornadoes occur uncertainly and short time, the most of tornadoes radar data are from a single Doppler radar on some kinds of mobile platforms. The moving either toward or away winds from the radar are showed by base velocity. In velocity images two kinds of color are used to denote the toward and away winds. Darker shading indicates slower winds, brighter colors indicate faster winds. When Tornado shows up in velocity images the two opposite winds color appear side by side with high brights [82]. Then the vortex is present as "couplet" in velocity image, so in CNNs process, the program actually are looking for the model described in figure 29 [83] and the features are presented those couplets like figure 30).

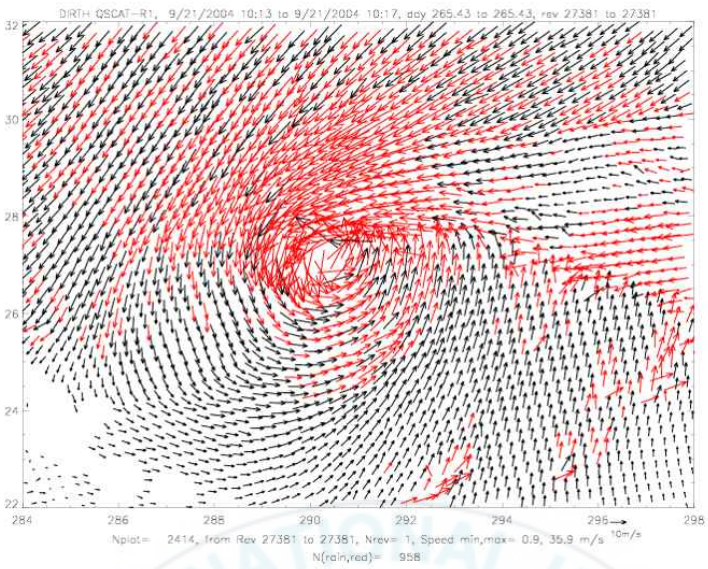


Fig. 28. The vortex in velocity image of Doppler radar

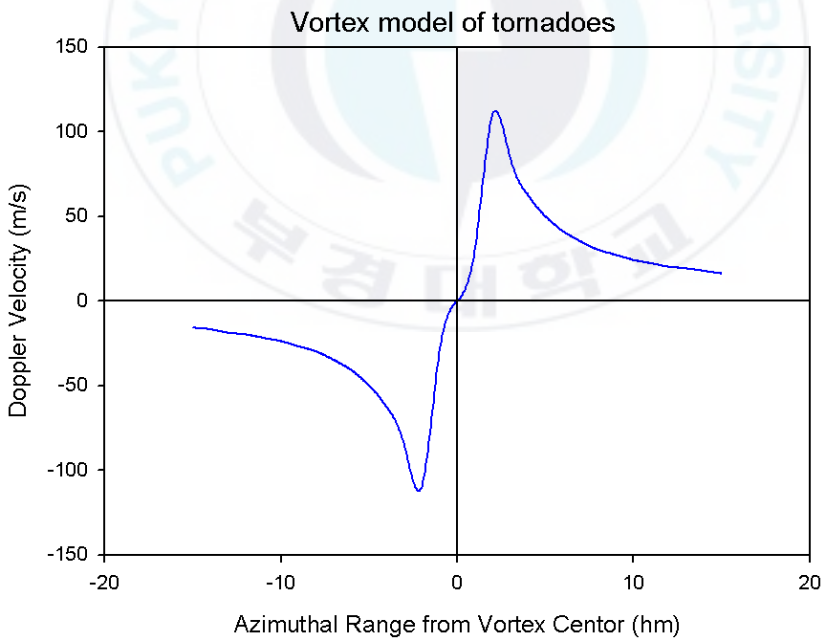


Fig. 29. Vortex model of tornadoes.

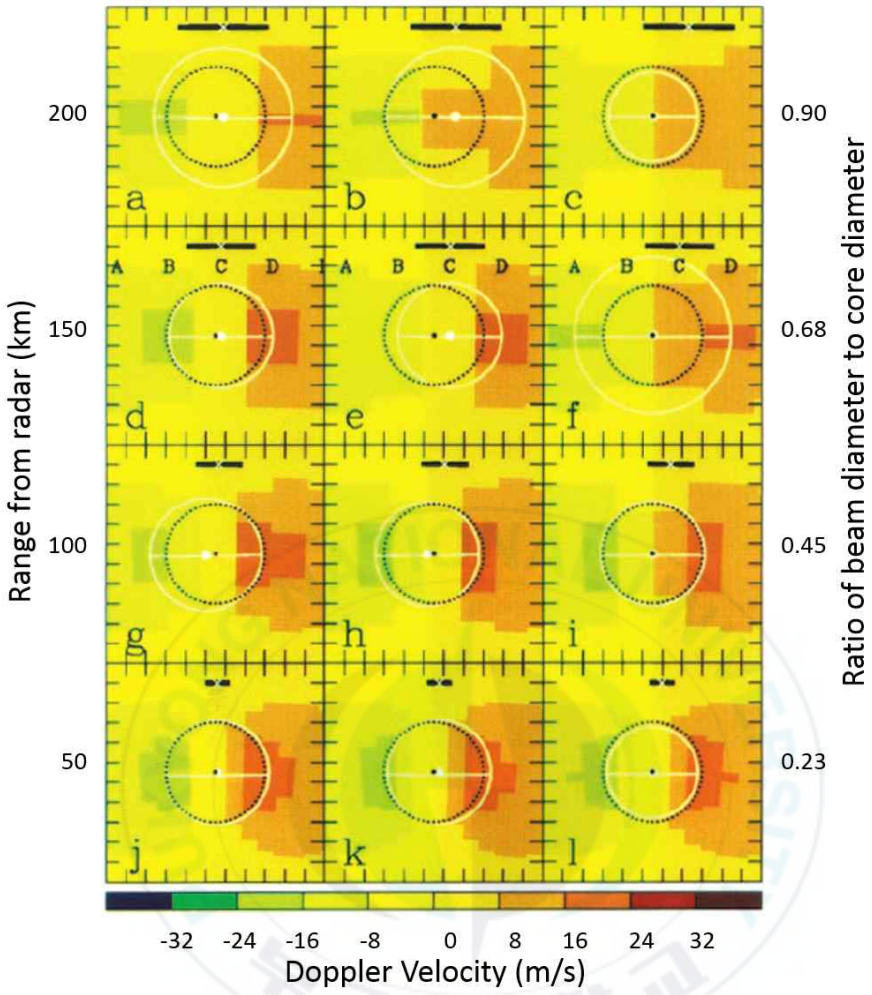


Fig. 30. The couplets of tornadoes in Doppler velocity images

Conclusion

While people focus on the short- time and fine scale weather disaster and the multi-application of weather information is used more and more. Tornadoes are the representative of them. For detecting tornadoes a fast radar data retrieval processing system is intense needed. On the hardware side people build a short-range dense radar network as solution, we attempt to make improvement on the software side. In this paper, we proposed a radar image retrieval process by contour-line of multi-layers based on MRF segmentation to generate radar images from a binary file. The major purpose is to avoid the most computationally expensive interpolation process by retrieving contour line. The proposed method contains nine steps in detail, from beginning to end they are raw file reading, data normalization, denoising, contour line abstraction, coordinate transformation, field interpolation, MRF segmentation, layers merging and CNNs tornadoes recognition. Through comparison between the conventional method and the proposed, the results show that the proposed method is more efficient. At last we analyzed the time distribution of each step, and the related factors for future improvement. In CNNs step, we analysised the features in tornadoes recognition process. According to experiments, the tornadoes can be recognized by CNNs method with fine tuning.

Appendix A
UFSXZ UF library user guide

Universal format tapes conform to the following five specification:

1. 9 track tapes, 1600 or 6250 cpi density. [Now extended to other media]
2. 16-bit words, signed integers, 2's complement.
   [Note that words are "big-endian" (most significant byte at end of word)]
3. Physical records, length <= 4095 words. [Record lengths now < 65536 bytes.]
4. File marks between volume scans (volumes).
5. ASCII words are left justified, blank filled.

MANDATORY HEADER BLOCK

Word

1 UF (ASCII)

2 Record length (16-bit words)

3 Position of first word of nonmandatory header block. (If no nonmandatory header block exists, this points to the first existing header block following the mandatory. In this way, word (3) always gives 1 + the length of the mandatory header.

4 Position of first word of local use header block. (If no local use headers exist, this points to the start of the data header block.)

5 Position of first word of data header block

6 Physical record number relative to beginning of file

7 Volume scan number relative to beginning of tape

8 Ray number within volume scan

9 Physical record number within the ray (one for the first

physical record of each ray)

10 Sweep number within this volume scan
11-14 Radar name (8ASCII characters, includes processor ID.)
15-18 Site name (8 ASCII characters)
19 Degrees of latitude (North is positive, South is negative)
20 Minutes of latitude
21 Seconds (x64) of latitude
22 Degrees of longitude (East is positive, West is negative)
23 Minutes of longitude
24 Seconds (x64) of longitude (Note: minutes and seconds have same sign as degrees.)
25 Height of antenna above sea level (meters)
26 Year (of data) (last 2 digits)
27 Month
28 Day
29 Hour
30 Minute
31 Second
32 Time zone (2 ASCII -- UT, CS, MS, etc.)
33 Azimuth (degrees x 64) to midpoint of sample
34 Elevation (degrees x 64)
35 Sweep mode: 0 - Calibration
1 - PPI (Constant elevation)
2 - Coplane
3 - RHI (Constant azimuth)
4 - Vertical
5 - Target (stationary)
6 - Manual
7 - Idle (out of control
36 Fixed angle (degrees x 64) (e.g., elevation of PPI; azimuth of RHI; coplane angle)

37 Sweep rate (degrees/seconds x 64)
38 Generation date of common format - Year
39 Month
40 Day
41-44 Tape generator facility name (8 character ASCII)
45 Deleted of missing data flag (Suggest 100000 octal)

OPTIONAL HEADER BLOCK

Word
1-4 Project name (8 ASCII)
5 Baseline azimuth (degrees x 64)
6 Baseline elevation (degrees x 64)
7 Hour (start of current volume scan)
8 Minute (start of current volume scan)
9 Second (start of current volume scan)
10-13 Field tape name (8 ASCII)
14 Flag (= 0 if number of range gates, R min, and spacing are the same for all data within this volume scan; = 1 if these are the same only within each sweep; = 2 if these are the same only within each ray).

LOCAL USE HEADER BLOCK

Any use, any contents

DATA HEADER

Word
1 Total number of fields this ray
2 Total number of records this ray

3 Total number of fields this record
4 1st field name (2 ASCII): e.g., VE - velocity (m/s)
SW - spectral width (m/s)
DM - reflected power dB(mW)
DZ - dB(Z)
etc.
5 Position of 1st word of 1st field header
6 2nd field name
7 Position of 1st word of 2nd field header
etc.

FIELD HEADER

Word
1 Position of first data word
2 Scale factor (meteorological units = tape value divided by scale factor)
3 Range to first gate (km)
4 Adjustment to center of first gate (m)
5 Sample Volume spacing (m)
6 Number of sample volumes
7 Sample volume depth (m)
8 Horizontal beam width (degrees x 64)
9 Vertical beam width (degrees x 64)
10 Receiver bandwidth (MHz)
11 Polarization transmitted (0 = horizontal; 1 =vertical; 2 = circular; >2 = elliptical)
12 Wavelength (cm x 64)
13 Number of samples used in field estimate
14 Threshold field (e.g., DM) (2 ASCII)
15 Threshold value
16 Scale

17 Edit code (2 ASCII)
18 Pulse repetition time (microseconds)
19 Bits per sample volume (16 for exchanged tape)
20-? Words for individual fields, as follows

for VF, VE, VR, VT, VP:
word
20 Nyquist velocity (scaled)
21 FL (2 ASCII) if flagged in least significant bit with
NCAR bad velocity flag (1 = good, 0 = bad)

for DM:
word
20 Radar constant = RC, such that dB(Z) = [(RC + DATA/SCALE]
+
20log(range in km)
21 Noise power (dB(mW) x scale)
22 Receiver gain (dB x scale)
23 Peak power (dB(mW) x scale)
24 Antenna gain (dB x scale)
25 Pulse duration (microseconds x 64)

Appendix B
Modified caffeNet structure

Training parts:

```
name: "CaffeNet"
state {
  phase: TRAIN
}
layer {
  name: "data"
  type: "Data"
  top: "data"
  top: "label"
  transform_param {
    mirror: true
    crop_size: 227
    mean_file:
"data_mine/image_mean_leveldb.bina
ryproto"
  }
  data_param {
    source:
"data_mine/train_leveldb"
    batch_size: 100
  }
}
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 96
    kernel_size: 11
    stride: 4
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
layer {
  name: "relu1"
  type: "ReLU"
```

```
    bottom: "conv1"
    top: "conv1"
  }
  layer {
    name: "pool1"
    type: "Pooling"
    bottom: "conv1"
    top: "pool1"
    pooling_param {
      pool: MAX
      kernel_size: 3
      stride: 2
    }
  }
  layer {
    name: "norm1"
    type: "LRN"
    bottom: "pool1"
    top: "norm1"
    lrn_param {
      local_size: 5
      alpha: 0.0001
      beta: 0.75
    }
  }
  layer {
    name: "conv2"
    type: "Convolution"
    bottom: "norm1"
    top: "conv2"
    param {
      lr_mult: 1
      decay_mult: 1
    }
    param {
      lr_mult: 2
      decay_mult: 0
    }
    convolution_param {
      num_output: 256
      pad: 2
      kernel_size: 5
      group: 2
      weight_filler {
        type: "gaussian"
        std: 0.01
      }
      bias_filler {
        type: "constant"
        value: 1
      }
    }
  }
  layer {
    name: "relu2"
    type: "ReLU"
    bottom: "conv2"
    top: "conv2"
  }
  layer {
    name: "pool2"
    type: "Pooling"
    bottom: "conv2"
    top: "pool2"
```

```
    pooling_param {
      pool: MAX
      kernel_size: 3
      stride: 2
    }
  }
  layer {
    name: "norm2"
    type: "LRN"
    bottom: "pool2"
    top: "norm2"
    lrn_param {
      local_size: 5
      alpha: 0.0001
      beta: 0.75
    }
  }
  layer {
    name: "conv3"
    type: "Convolution"
    bottom: "norm2"
    top: "conv3"
    param {
      lr_mult: 1
      decay_mult: 1
    }
    param {
      lr_mult: 2
      decay_mult: 0
    }
    convolution_param {
      num_output: 384
      pad: 1
      kernel_size: 3
      weight_filler {
        type: "gaussian"
        std: 0.01
      }
      bias_filler {
        type: "constant"
        value: 0
      }
    }
  }
  layer {
    name: "relu3"
    type: "ReLU"
    bottom: "conv3"
    top: "conv3"
  }
  layer {
    name: "conv4"
    type: "Convolution"
    bottom: "conv3"
    top: "conv4"
    param {
      lr_mult: 1
      decay_mult: 1
    }
    param {
      lr_mult: 2
      decay_mult: 0
    }
    convolution_param {
```

```
    num_output: 384
    pad: 1
    kernel_size: 3
    group: 2
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 1
    }
  }
}
layer {
  name: "relu4"
  type: "ReLU"
  bottom: "conv4"
  top: "conv4"
}
layer {
  name: "conv5"
  type: "Convolution"
  bottom: "conv4"
  top: "conv5"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 256
    pad: 1
    kernel_size: 3
    group: 2
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 1
    }
  }
}
layer {
  name: "relu5"
  type: "ReLU"
  bottom: "conv5"
  top: "conv5"
}
layer {
  name: "pool5"
  type: "Pooling"
  bottom: "conv5"
  top: "pool5"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
```

```
}
layer {
  name: "fc6"
  type: "InnerProduct"
  bottom: "pool5"
  top: "fc6"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  inner_product_param {
    num_output: 4096
    weight_filler {
      type: "gaussian"
      std: 0.005
    }
    bias_filler {
      type: "constant"
      value: 1
    }
  }
}
layer {
  name: "relu6"
  type: "ReLU"
  bottom: "fc6"
  top: "fc6"
}
layer {
  name: "drop6"
  type: "Dropout"
  bottom: "fc6"
  top: "fc6"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layer {
  name: "fc7"
  type: "InnerProduct"
  bottom: "fc6"
  top: "fc7"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  inner_product_param {
    num_output: 4096
    weight_filler {
      type: "gaussian"
      std: 0.005
    }
    bias_filler {
      type: "constant"
      value: 1
    }
```

```
    }
  }
  layer {
    name: "relu7"
    type: "ReLU"
    bottom: "fc7"
    top: "fc7"
  }
  layer {
    name: "drop7"
    type: "Dropout"
    bottom: "fc7"
    top: "fc7"
    dropout_param {
      dropout_ratio: 0.5
    }
  }
  layer {
    name: "fc_mine"
    type: "InnerProduct"
    bottom: "fc7"
    top: "fc_mine"
    param {
      lr_mult: 10
      decay_mult: 1
    }
    param {
      lr_mult: 20
      decay_mult: 0
    }
    inner_product_param {
      num_output: 2
      weight_filler {
        type: "gaussian"
        std: 0.01
      }
      bias_filler {
        type: "constant"
        value: 0
      }
    }
  }
  layer {
    name: "loss"
    type: "SoftmaxWithLoss"
    bottom: "fc_mine"
    bottom: "label"
  }
```

Reference

[1] K. M. Ban. (2014, Mar.). The 2015 Global Assessment Report on Disaster Risk Reduction. The United Nations Office for Disaster Risk Reduction. [Online]. Available:
http://www.preventionweb.net/english/hyogo/gar/2015/en/home/download.html

[2] P. L. Smith, *Weather Radar Technology beyond NEXRAD*. National research council, National Academies Press, 2002.

[3] D. Mc-laughlin, D. Pepyne, and B. Philips et al., "Short-wavelength technology and the potential for distributed networks of small radar system," *Bulletin of the American Meteorological Society*, vol. 90, pp. 1797-1817, Dec. 2009.

[4] J. Brotzge, S. Erickson, and H. Brooks, "A 5-yr climatology of tornado false alarms," *Weather and Forecasting*, vol. 26, pp. 534-544, Aug. 2011.

[5] V. Kambhampaty, R. Gali, and N. Prasad, "A short term tornado prediction model using satellite imagery," *IEEE International Conference on Systems Informatics, Modelling and Simulation*, pp. 132-138, May. 2014.

[6] [Online]. Available:
http://www.nssl.noaa.gov/education/svrwx101/tornadoes/

[7] G. E. Hinton, S. Osindero, and Y. The, "A fast learning algorithm for deep belief nets," *Neural Computation,* pp.1527-1554, 2006.

[8] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy Layer-Wise Training of Deep Networks," *Advances in Neural Information Processing Systems 19 (NIPS 2006)* pp. 153-160, 2007.

[9] M. A. Ranzato, C. Poultney, S. Chopra, and Y. Lecun, " Efficient learning of sparse representations with an energy-based model," *Advances in Neural Information Processing Systems (NIPS 2006)*, pp.1137-1144, 2007.

[10] B. A. Olshausen, P. Sallee, and M. S. Lewicki, "Learning sparse image codes using a wavelet pyramid architecture," *Advances in Neural*

Information Processing Systems, pp.887-893, 2001.
[11] M. S. Lewicki and B. A. Olshausen, "Probabilistic framework for the adaptation and comparison of image codes," *Journal of the Optical Society of America*, pp.1587-1601, 1999.
[12] S. Lawrence, G. L. Giles, and A. D. Back, "Face recognition: a convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no.1, pp. 98-113, Jan. 1997.
[13] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning,* pp.1-127, 2009.
[14] R. J. Doviak and D. S. Zrnic, *Doppler radar and Weather Observations*, 2nd ed. San Diego CA: Academic Press, 1993.
[15] E. R. Hilgendorf and R. H. Johnson, "A study of the evolution of mesoscale convective systems using wsr-88d data," *Weather Forecasting*, vol. 13, no. 2, pp. 437-452, 1998.
[16] M. A. Richards, *Fundamentals of Radar Signal Processing*, 2nd ed. New York, NY: McGraw-Hill, 2005.
[17] J. Zhang, K. Howard, W. Xia, and J. J. Gourley, "Comparison of objective analysis schemes for the WSR-88D radar data," *31th International Conference on Radar Meteorology*, pp. 907-910, Aug. 2003.
[18] W. K. Jenkins, B. Mather, and D. C. Munson, " Nearest neighbor and generalized inverse distance interpolation for Fourier domain image reconstruction," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 1069-1072, Apr. 1985.
[19] R. J. Trapp and C. A. Doswell, "Radar Data Objective Analysis," *Journal of Atmospheric and Oceanic Technology*, vol. 17, pp. 105-120, Feb. 2000.
[20] S. F. Liu and S. H. Shen, "An adaptive image interpolation method focusing on edge information," *International Symposium on Computational Intelligence and Design*, vol. 1, pp. 421-424, Dec. 2014.
[21] M. Panggabean, O. Tamer, and L. A. Ronningen, "Parallel image transmission and compression using windowed kriging interpolation," *IEEE International Symposium on Signal Processing and Information*

Technology, pp. 315-320, Dec. 2010.
[22] M. A. Askelson, J. P. Aubagnac, and J. M. Straka, “An adaptation of the barnes filter applied to the objective analysis of radar data,” *Monthly Weather Review*, vol. 128, no. 9, pp. 3050-3082, Sep. 2000.
[23] S. L. Barnes, "A technique for maximizing details in numerical weather map analysis," *Journal of Applied Meteorology*, vol. 3, pp. 396-409, Aug. 1964.
[24] Y. Kuleshov, G. de Hoedt, W. Wright, and A. Brewster, "Thunderstorm distribution and frequency in Australia," *Australian Meteorological Magazine*, vol. 51, pp. 145-154, Sep. 2002.
[25] P. M. Markowski, "Hook echoes and rear-flank downdrafts: a review," *Mon. Wea. Rev.*, pp. 852-876, 2002.
[26] D. W. Burgess, M. A. Magsig, J. Wurman, D. C. Dowell, and Y. Richardson, “Radar observations of the 3 May 1999 Oklahoma city tornado,” *Weather and Forecasting,* vol. 17, pp. 456-471, Jun. 2002.
[27] A. Jim, "ISWS is pioneer in tracking tornadoes by radar," Apr. 2013
[28] *Tornado warning guidance*, National Weather Service. spring 2013.
[29] H. K. Wang, R. E. Mercer, J. L. Barron, and P. Joe, "Skeleton based tornado hook echo detection," *International Conference on Image Processing*, vol. 6, pp. 361-364, 2007.
[30] G. S. Forbes, "On the reliability of hook echoes as tornadoes indicators," *Mon. Weather Rev.*, pp.1457-1466, 1981.
[31] [Online]. Available:
http://www.srh.noaa.gov/jan/?n=2010_04_24_tor_radarimagery
[32] National Oceanic and Atmospheric Administration, "Enhanced F scale for tornado damage," Storm Prediction Center. [Online]. Available:
http://www.spc.noaa.gov/efscale/ef-scale.html
[33] A. J. Thomas, M. M. Kevin, and T. S. John, "Association between NSSL mesocyclone detection algorithm-detected vortices and tornadoes," *Weather and Forecasting*, vol. 19, pp.872-890, Oct. 2004.
[34] B. T. Frank, R. K. Kevin, and J. V. Stanley, "Tornado detection based on seismic signal," *Journal of Applied Meteorology,* vol. 34, pp. 572-582,

Jul. 1994.
[35] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *J. Physiol.,* vol. 195, no. 1, pp.215-243, Mar. 1968.
[36] K. Fukushima, "Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.,* vol. 36, no. 4, pp.193-202, Apr. 1980.
[37] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE,* vol. 86, no. 11, pp. 2278-2324, Nov. 1998.
[38] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, pp. 1106-1114, 2012.
[39] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification," *Proc. IJCNN*, pp. 1918-1921, 2011.
[40] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German traffic sign recognition benchmark: a multi-class classification competition," *Proc. IJCNN*, pp. 1453-1460, 2011.
[41] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. FeiFei, "Large-scale video classification with convolutional neural networks," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1725-1732, Jun. 2014.
[42] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: integrated recognition, localization and detection using convolutional networks," *International Conference on Learning Representations*, pp. 1-15, 2013.
[43] C. Szegedy, W. Liu, Y. Q. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1-9, Jun. 2015.
[44] S. X. Zhang, "CASA real-time multi-Doppler retrieval system," M.S.

thesis, Colorado State University, Fort Collins, CO, Jun. 2011.
[45] S. L. Barnes, “Report on a Meeting to Establish a Common Doppler Radar Data Exchange Format,” *The Bulletin of the American Metrological Society*, vol. 61, no. 11, pp. 1401-1404, Nov. 1980.
[46] *IRIS programmer's manual*, Vaisala Oyj, Inc., Helsinki, Finland, 2013.
[47] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one" *Society for industrial and applied mathematics*, vol. 4, pp. 490-530, Apr. 2005.
[48] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679-698, Nov. 1986.
[49] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics and Image Processing*, vol. 30, pp. 32-46, Apr. 1985.
[50] G. Bradski and A. Kaehler, *Learning OpenCV*, USA: O'Reilly Media, 2008.
[51] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721-741, Nov. 1984.
[52] J. Besag, "Spatial interaction and the statistical analysis of lattice systems," *Journal of the Royal Statistical Society. Series B*, vol. 36, no. 2, pp. 192-236, 1974.
[53] J. M. Hammersley and P. Clifford, "Markov fields on finite graphs and lattices," 1971 (unpublished).
[54] P. Clifford, "Markov random fields in statistics," in Disorder in Physical Systems: A Volume in Honour of John M. Hammersley, G. Grimmett and D. Welsh, Ed. New York: Oxford University Press, 1990, pp. 19-32.
[55] K. Binder, "Ising model," in Encyclopaedia of Mathematics, Hazewinkel, Michiel, Ed. New York: Springer Publishers. 2001.
[56] C. Hongler and S. Smirnov, "The energy density in the planar Ising model," *Acta Mathematica*, vol. 211, pp. 191-225, Dec. 2013.

[57] M. Arbib, *The handbook of brain theory and neural networks*, Cambridge MA: The MIT Press. 1995.
[58] I. Wegener, "Simulated Annealing Beats Metropolis in Combinatorial Optimization," in Automata, Languages and Programming, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Ed. Berlin Heidelberg: Springer-Verlag, 2005, pp. 589-601.
[59] P. Heckbert, "A seed fill algorithm," in Graphics Gems, A. S. Glassner, Ed. San Diego. CA: Academic Press, 1990. pp. 275-277.
[60] J. R. Shaw. (2004, Mar.). QuickFill: An Efficient Flood Fill Algorithm. [Online]. Available:
http://www.codeproject.com/gdi/QuickFill.asp
[61] E. Ruzanski, V. Chandrasekar, and Y. Wang, "The CASA nowcasting system," *Journal of Atmospheric and Oceanic Technology*, vol. 28, no. 5, pp. 640-655, May 2011.
[62] N. Z. Feng, L. Y. Ma, and Y. Shen, "Fuzzy partition based curvelets and wavelets denoise algorithm," *Computational Intelligence and Security Workshops*, pp. 23-26, Dec. 2007.
[63] Z. Huang, "A median filter based on judging impulse noise by statistic and adaptive threshold," *Image and Signal Processing*, vol. 3, pp. 207-210, May. 2008.
[64] R. Sluiter, "Interpolation methods for climate data literature review," Royal Netherlands Meteorological Institute, De Bilt Netherlands, IR. 2009-04 KNMI, 2009.
[65] R. Kvasov, S. Cruz-Pol, J. Colom-Ustariz, L. Leon Colon, and P. Rees, "Weather radar data visualization using first-order interpolation," *IEEE International on Geoscience and Remote Sensing Symposium*, pp. 3574-3577, July 2013.
[66] M. I. Skolnik, *Introduction to Radar System*, 3th ed. New Delhi, ND: McGraw-Hill, 2001.
[67] M. A. Richards, *Fundamentals of Radar Signal Processing*, New York, NY: McGraw-Hill, 2005.
[68] D. Flores-Tapia, G. Thomas, and S. Pistorius, "A comparison of

interpolation methods for breast microwave radar imaging," *Engineering in Medicine and Biology Society*, pp. 2735-2738, Sept. 2009.
[69] K. Shinozawa, M. Fujii, and N. Sonehara, "A weather radar image prediction method in local parallel computation," *IEEE World Congress on Computational Intelligence Neural Networks*, vol. 7, pp. 4210-4215, July 1994.
[70] S. L. Barnes, "Report on a meeting to establish a common Doppler radar data exchange format," *The Bulletin of the American Metrological Society*, vol. 61, no. 11, pp. 1401-1404, Nov. 1980.
[71] J. Zhang, H. Ken, W. W. Xia, and J. J. Gourley, "Comparison of objective analysis schemes for the WSR-88D radar data," *31th International Conference on Radar Meteorology*, Aug. 2003.
[72] R. Kvasov, S. Cruz-Pol, J. Colom-Ustariz, L. Leon Colon, and P. Rees, "Weather radar data visualization using first-order interpolation," *2013 IEEE International on Geoscience and Remote Sensing Symposium*, pp. 3574-3577, July 2013.
[73] M. A. Askelson, J. P. Aubagnac, and J. M. Straka, "An adaptation of the barnes filter applied to the objective analysis of radar data," *Monthly Weather Review*, vol. 128, no. 9, pp. 3050-3082, 2000.
[74] C. Marzban, and G. J. Stumpf, "A neural network for damaging wind prediction," *Weather and Forecasting,* vol. 16, no. 5, pp.600-610, 1998.
[75] T. B. Trafalis, I. Adrianto, and M. B. Richman, "Active learning with support vector machines for tornado prediction," *International Conference on Computational Science ICCS 2007*, pp. 1130-1137, 2007.
[76] A. V. Ryzhkov, T. J. Schuur, D. W. Burgess, and D. S. Zrnic, "Polarimetric tornado detection," *Jornal of Applied Meteorology and Climatology*, vol. 44, no.5, pp.557-570, May. 2005.
[77] C. Marzban and G. J. Stumpf, "A neural network for tronado prediction based on Doppler radar-derived attributes," *Journal of Applied Meteorology and Climatology*, vol. 35, no.5, pp.617-626, May. 1996.
[78] R. A. Brown, L. R. Lemon, and D. W. Burgess, "Tornado detection by pulsed Doppler radar," *Monthly Weather Review*, pp. 29-38, Jan. 1978.

[79] V. Lakshmanan, I. Adrianto, T. Smith, and G. Stumpf, "A spatiotemporal approach to tornado predication," *IEEE International Joint Conference on Neural Networks*, vol. 3, pp.1642-1647, 2005.
[80] V. T. Wood and R. A. Brown, "Sampling strategies for tornado and mesocy clone detection using dynamically adaptive Doppler radar a simulation study," *Journal of Atmospheric and Oceanic Technology*, vol. 26, pp. 492-507, Mar. 2009.
[81] F. Junyent, S. Frasier, D. J. McLaughlin, and V. Chandrasekar, "High resolution dual-polarization radar observation of tornados: implications for radar development and tornado detection," *IEEE International Geoscience and Remote Sensing Symposium*, vol.3, pp. 2034-2037, 2005.
[82] E. D. Mitchell, S. V. Vasiloff, G. J. Stumpf, A. Witt, M. D. Eilts, J. T. Johnson, and K. W. Thomas, "The national severe storms laboratory tornado detection algorithm," Weather and Forcasting, vol. 13, pp. 352-366, June 1998.
[83] V. T. Wood and R. A. Brown, "Effects of radar sampling on single-Doppler velocity signatures of mesocyclones and tornadoes," *Weather and Forecasting*, vol. 12, no. 4, pp. 928-938, Dec. 1997.

Acknowledgement