11A.1 EFFICIENT REDUCTION AND COMPRESSION OF WEATHER RADAR DATA IN UNIVERSAL FORMAT

W. David Pan*
Dept. of Electrical & Computer Engineering, University of Alabama in Huntsville, Huntsville, AL

Paul R. Harasti
Visiting Scientist Programs, University Corporation for Atmospheric Research, Boulder, CO

Michael Frost, Qingyun Zhao, John Cook
Marine Meteorology Division, Naval Research Laboratory, Monterey, CA

Timothy Maese
Basic Commerce and Industries, Moorestown, NJ

Lee J. Wagner
Atmospheric Propagation Branch, SPAWAR Systems Center, San Diego, CA

1. INTRODUCTION

The Naval Research Laboratory (NRL) Marine Meteorology Division, Monterey, CA is developing a weather radar data assimilation system to enhance the safety of ship and at-sea aircraft operations. The system will take advantage of Navy vessels in the battle fleet having weather radar capability. These radars execute full-resolution volume scans and have signal processors that both display and archive the weather data. The data are archived in files following the so-called universal format (UF) as defined in Barnes (1980).

NRL is working with Space and Naval Warfare Systems Center (SPAWAR), San Diego, CA on their development of a system to transmit the UF files created by the Hazardous Weather Detection and Display Capability (HWDDC) system onboard Navy ships to Fleet Numerical Meteorology and Oceanography Center (FNMOC) in Monterey, CA. The plan is to incorporate weather radar data into the Navy's Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS®)[1] forecast model and various nowcasting algorithms (Cook et al. 2007; Zhao et al. 2005). The products from these nowcasts and forecasts systems would then be made available to all US defense forces through the FNMOC net-centric information system.

In January 2006, a prototype HWDDC was successfully tested with a land-based SPS-48E radar at Navy facilities in Dam Neck, VA (Harasti et al. 2006; Maese et al. 2007). Later in February of 2006, the HWDDC was deployed onboard the USS PELELIU (LAH5) for a 6-month, at-sea demonstration. The SPS-48E is a S-band, long range, air defense, volume scanning radar onboard US Navy aircraft carriers and large-deck amphibious ships. It operates with multiple pencil beams in a mechanically rotating phased-array antenna that scans electronically in elevation, and completes a volume scan in 4 seconds (Harasti et al. 2007). A Weather Extractor Computer (WEC) and a Weather Data Interface Card (WDIC) have been developed for the SPS-48E that provide Doppler data at the lowest three elevation scans and reflectivity data at all elevation scans in Universal Format (UF). The WEC and WDIC are part of the HWDDC system that provides real-time weather information to shipboard personnel. The HWDDC system taps into the SPS-48E radar returns without changing or interfering with the tactical target scanning configuration. See Maese et al. (2007) for more information on the SPS-48E and the HWDDC.

To minimize the load on the operational bandwidth, the UF files have to be significantly com-

*Corresponding author address: W. David Pan, Dept. of Electrical and Computer Engineering, University of Alabama in Huntsville, Huntsville, AL 35899. Tel: (256) 824-6642, Email: dwpan@eng.uah.edu.

[1]COAMPS® is a registered trademark of the Naval Research Laboratory.

pressed before transmission off the ship to a shore location, requiring a high compression rate that even the state-of-the-art data compressors cannot provide.

In the literature, there has been fairly limited prior work on compression of weather radar data in UF format. In Makkapati and Mahapatra (2007), a method for achieving high levels of compression on weather radar data was presented. However, the compression was not on original radar data, but on the weather reflectivity contours (which were treated as 2D images). Very high compression was achieved at the expense of perceptible loss of meteorological information. Nevertheless, the user cannot control where the loss of meteorological information would come from. Therefore, such an image-oriented method would not be suitable for our problem, where a forecast model or nowcasting algorithm will be the ultimate consumer of the reconstructed radar data.

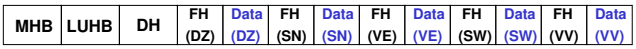
On the other hand, Kruger and Krajewski (1997) described a compression and archiving strategy for weather radar data based on *run length coding*. Data reduction was also used by coding only reflectivity data above a certain signal-to-noise ratio. Results was presented for 62 days of reflectivity data from a WSR-88D radar operated by the United States National Weather Service (NWS), as well as for 60 days of reflectivity data for a radar operated by the Bureau of Meteorology in Australia. The compression and reduction algorithms proposed in Kruger and Krajewski (1997) were able to reduce the two data sets from the original size of 60 GB to 4.8 GB, thereby achieving about 13 to 1 compression.

This paper presents a software package for compressing UF files which contain not only reflectivity data, but also other types of data fields as well (see Table 1). We introduce more efficient methods to compress record headers, and we employ more aggressive data reduction via thresholding operations, while still maintaining sufficient information content to impact analyses and forecasts at FNMOC with COAMPS®. In addition, we have developed UF file compression software that builds on widely used data compression utilities such as *bzip2*, which employs much more sophisticated data compression methods than the simple *run length coding* method.

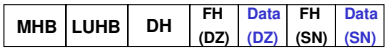
2. REQUIREMENTS OF UF FILE COMPRESSION

2.1. *SPS-48E UF Files*

For the purposes of radar data assimilation, three UF files per hour will be transmitted to FNMOC in near-real-time. To minimize the load on the operational bandwidth, these large UF files have to be significantly compressed to below 1 MB before transmission. In an effort to develop efficient weather radar data reduction and compression techniques to meet this requirement, we analyzed archived SPS-48E UF data obtained from an at-sea experiment onboard the USS PELELIU (LHA5) in February 2006. This data set contains a wide range of precipitation echoes spanning 21 hours of observations. There were a total of 252 UF files in the data set, each of size 5.4 MB, containing data taken from a total of 22 fixed elevation angles (tilts). The data from the tilts each contain 360 azimuthal rays (ranging from 1° to 360°), and are stored in 360 records. Each record contains the headers as depicted in Table 1. Figure 1 shows how headers and data are interleaved in a record. Note the difference in header sizes between records located at the lower three elevations and those at upper elevations.



(a)



(b)

Figure 1: Structure of records in UF files. The structure at (a) the lower 3 elevations, and (b) the upper 19 elevations in a volume scan. See Table 1 for more details.

2.2. *The UF File Compression Challenge*

Our goal is to compress each UF file to below 1 MB to meet the constraint on bandwidth usage, which can be translated to a minimum compression factor of over **5** to 1. To compress these UF files, we tested three well-known lossless data compression utilities, including *gzip*[2], *bzip2*[3], and *lpaq8*, which is a "lite" version of *PAQ*[4], a series of data compression archivers that have achieved top rankings on

[2] http://www.gzip.org
[3] http://www.bzip.org
[4] http://www.cs.fit.edu/~mmahoney/compression/

Table 1: Headers in a record (n is the number of data fields in the record).

Header Name	Abbreviation	Size (in 2-byte word)
Mandatory Header Block	MHB	45
Local Use Header Block	LUHB	9
Data Header	DH	$3 + 2 \times n$
Field Header (Reflectivity)	FH (DZ)	19
Field Header (Signal-to-Noise Ratio)	FH (SN)	19
Field Header (Radial Velocity)	FH (VE)	21
Field Header (Spectrum Width)	FH (SW)	21
Field Header (Valid Velocity Flag)	FH (VV)	21

several benchmarks measuring compression ratio, such as the Hutter Prize and the Calgary Challenge, albeit at the expense of speed and memory usage. Unlike *PAQ*, *lpaq8* is a single file compressor, and it runs faster than *PAQ* but with less compression.

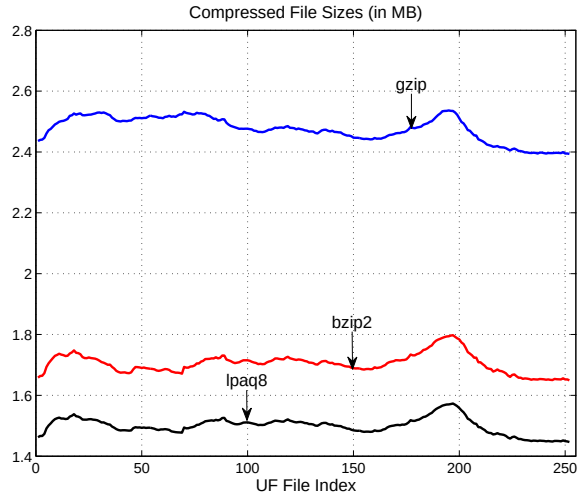


Figure 2: The sizes of the compressed UF files with three lossless data compression utilities. Average compression factors are: 2.19:1 (*gzip*), 3.18:1 (*bzip2*), and 3.61:1 (*lpaq8*).

The tests were performed on a PC (with Intel® Core™2 Duo Desktop Processor E4600, 2.39 GHz, 1 GB of RAM, and Windows XP Professional Version 2002, Service Pack 3). For both *gzip* and *bzip2*, option 9 was chosen to maximize the compression. *lpaq8* took noticeably long time to finish than the other two utilities. For example, *lpaq8* (with option 6) compressed the UF file (taken at 00:02:10 UTC) from 5,680,800 bytes to 1,533,379 bytes in about 11 seconds, using 198 MB of memory. Even though the state-of-the-art *PAQ* utility (version *paq8p* with option 5) was able to compress the same UF file to an even smaller size (about 1.3 MB), the compression took 14 minutes and 0.2 GB of memory, thereby rendering *PAQ* an unacceptable choice for near-real-time applications, where one UF file is generated every 5 minutes. In terms of compressed file sizes, neither *PAQ* nor the other three utilities (Figure 2) could meet our need to reduce the UF file size to below 1 MB.

3. UF FILE COMPRESSION AND REDUCTION ALGORITHMS

In an effort to overcome the challenge of UF file compression, we employed a divide-and-conquer approach on these UF files by separating the actual radar data from the headers, as shown in Figure 3. Since the headers were found to account for about 30% of the overall UF file size (Figure 4), significant compression on headers would be critical to achieving high compression on the entire UF file.

3.1. *Header Compression*

Parameters such as the radar name, location, volume date, sweep mode, and so on are archived as part of the record headers. Since many of these parameters tend not to change from record to record, the headers contain a substantial amount of redundancy. For example, in the UF files considered in our study, the 45-word mandatory header blocks of the first and second records differ in only 4 words, including word 6 (physical record number relative to beginning of file), word 8 (ray number within volume scan), word 33 (azimuth), and word 36 (fixed angle). Similarly large amount of redundancy was also observed for local use header blocks, data headers and field headers of neighboring records in UF files. Large amount of redundancy existing in headers would be better exploited by data compression utilities such as *bzip2*, if headers were not inter-

leaved with data, as in original UF files. Figure 3(a) shows how a macro-header is formed by concatenating all the headers in a record at the lower three elevations. In a similar fashion, a large data block can be constructed by merging all the data of different types (Figure 3(b)). Note that for the upper 19 elevations, macro-headers will be shorter due to the absence of field headers for VE, SW, and VV data (see Figure 1). As a result of this reorganization of headers and data, there will be one macro-header and one data block for each record. The macro-headers for all the records account for 30% of the overall UF file sizes (Figure 4), highlighting the importance of achieving high compression on headers.

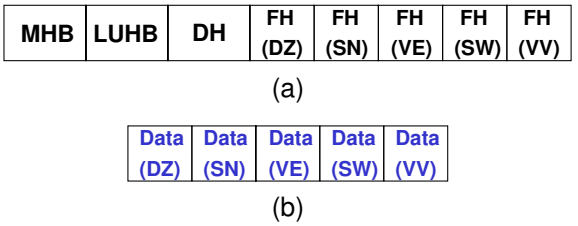


Figure 3: (a) Macro-header formed by merging all the headers in a record. (b) A block of data formed by merging all the data fields.

Due to significant redundancy between macro-headers of neighboring records, compression would be made more efficient by coding the difference (or update) between successive macro-headers than coding directly the macro-headers themselves. This approach is generally known as DPCM (Differential Pulse Code Modulation), and it has enjoyed wide applications in image and video signal compression. In conventional DPCM approaches, the differential operation takes the form

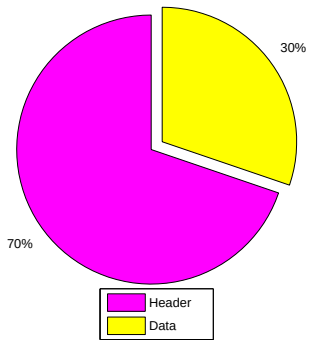


Figure 4: The breakdown of sizes of headers (1,717,200 bytes) and data (3,963,600 bytes) in an UF file.

of subtraction. However, in the special context of UF headers, subtraction between two 16-bit words (treated as signed numbers) may potentially yield a result that requires more than 16 bits in representation. Truncation of such a result to a 16-bit word will introduce errors. Since strictly lossless compression on headers is of utmost importance, to avoid the potential problem of dynamic range being exceeded, we proposed to use bitwise XOR $\oplus$ (Exclusive OR), an alternative operation that tends to be "safer" and faster than subtraction, to obtain the differences between neighboring macro-headers. Given two neighboring macro-headers H_i and H_{i+1}, the difference between them can be found as

$$D_i = H_i \oplus H_{i+1}. \quad (1)$$

If two co-located words h_i ($\in H_i$), and h_{i+i} ($\in H_{i+1}$) are identical, then $h_i \oplus h_{i+1} = 0$. Hence if two neighboring macro-headers differ in just a very small number of words, the difference D_i will be a very sparse vector of words with most of the words being zero.

Given a perfectly reconstructed macro-block H_i, and the differential block D_i (update), the succeeding macro-block H_{i+1} can be recovered by

$$H_i \oplus D_i = H_i \oplus (H_i \oplus H_{i+1}) = H_{i+1}. \quad (2)$$

As shown in Figure 5, applying XOR operations repeatedly generates differences between neighboring macro-headers. At the decoder, based on the first macro-block (H_0) and the differences D_i, reconstruction of the remaining macro-headers can be accomplished by applying equation (2) repeatedly. As shown in Figure 1 and Table 1, record headers in the lower three elevations are different from those in the upper elevations. Therefore, the macro-headers in records at the lower three elevations have a size of 168 ($= 45 + 9 + 13 + 19 \times 2 + 21 \times 3$) words, whereas the size of macro-headers in records at the upper elevations is 99 ($= 45 + 9 + 7 + 19 \times 2$) words. In Figure 5, the macro-headers in the upper 19 elevations were zero-padded so that the XOR-based DPCM would be applied on all macro-headers with the uniform size of 168 words, without any impact on the compression efficiency. Figure 6 shows an example of XOR-based DPCM applied on an UF file, where differential macro-headers contain fewer non-zero entries than original headers. While *bzip2* compressed the macro-headers to about 8 KB, the differential macro-headers can be compressed to only 2.5 KB, leading to approximately 3 times more efficient compression.

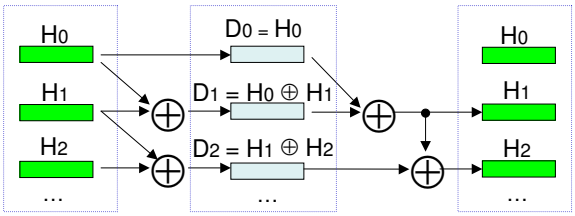


Figure 5: Generation of differential macro-headers and reconstruction of original macro-headers through XOR operations.

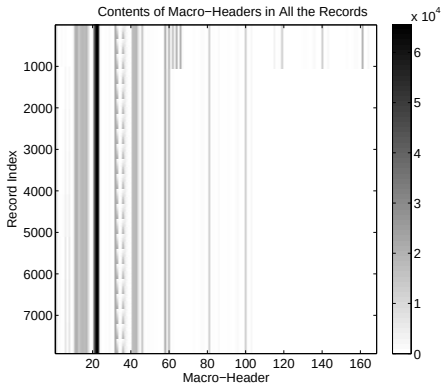


Figure 6: (a) Content of macro-headers in the UF file with time stamp 00:02:10 UTC. (b) Differential macro-headers obtained by applying the XOR-based DPCM on the macro-header block in (a). There are a very small number of non-zero entries (dark dots).

Figure 7 summarizes the result of losslessly compressing headers by using the *bzip2* utility in 252 UF files in the data set. On average, headers in each UF file can be compressed from about 1,677 KB to about 2.5 KB, achieving an average compression factor of over **600** to 1.

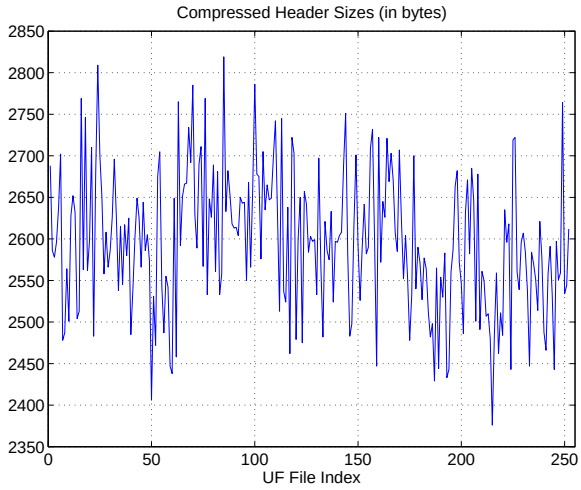


Figure 7: Compressed header sizes (average 2.5 KB).

3.2. *Data Reduction and Compression*

Now that headers can be squeezed into negligibly small sizes, we can focus on compression of radar data. Once all the data have been merged into a large block, as shown in Figure 3, the data block can be compressed by using a compression utility such as *bzip2*. In actual implementation, prior to compression, the original data (tape values) are divided by the scaling factors indicated in field headers to obtain the actual data in meteorological units (Barnes 1980). The scaled down values are then rounded to the nearest integers to facilitate compression. To reconstruct the original tape data (tape values), the reverse process is carried out. Since rounding errors are introduced, the compression is not strictly lossless. However, these very small rounding errors (with absolute values less than 0.5) were negligible and testing shows they do not significantly impact the numerical models following decompression. Figure 8 shows that data in an UF file can be compressed by *bzip2* from 3,963,600 bytes per file to 829 KB on average. Given that the headers per UF file can be compressed to 2.5 KB on average (see Figure 7), the UF files can be compressed down to below 1 MB, thereby accomplishing our goal.

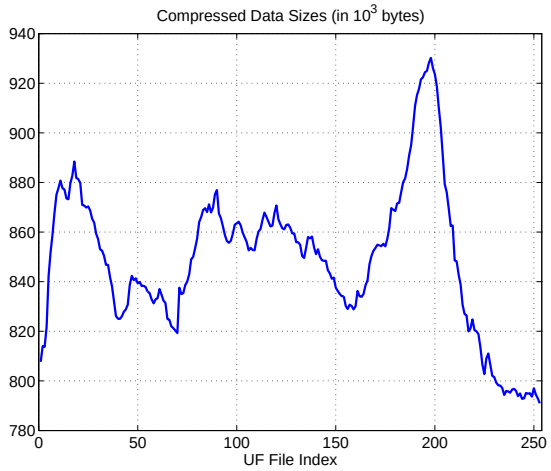


Figure 8: Compressed data sizes by using *bzip2*. Data in an UF file can be compressed to 829 KB on average.

In addition to attaining significant data compression, further reduction on the original data is also possible with our specific applications. For example, the SN (Signal-to-Noise Ratio) data were collected primarily for quality control purposes, hence they do not have to be transmitted if quality control is accomplished onboard ship. To further reduce the load on bandwidth, it was decided that only three types of data in the original UF files need to be transmitted, including DZ (reflectivity), VE (radial velocity), and SW (spectrum width). Furthermore, thresholding was introduced to blank out data whose values are below certain thresholds as determined by the appropriate quality control requirements. For example, if a location has either DZ $\leq$ 5 dBZ or SN $\leq$ 10 dB, then all three types of co-located data, if available, will be blanked out and then labeled with an identical marker. Thresholding creates long runs of identical values for the blank-out marker, leading to more efficient data compression.

3.3. *UF File Compression Software*

A software package was developed to achieve UF file compression and decompression. The flowcharts for the encoder and the decoder are shown in Figure 9. At the encoder, a file reader extracts the macro-headers and the data block from the original UF file. XOR-based DPCM is then applied on the macro-headers to obtain the differential macro-headers. In the meantime, thresholding is optionally applied to the data block, with thresholds being fully configurable. Next, the differential headers and data will be compressed based on an open-source implementation of the *bzip2* utility. The compressed bit stream will be transmitted over the channel. At the decoder, the compressed bit stream will be decoded by *bzip2*, thereby recovering the differential macro-headers and the data block. After the original headers are reconstructed by the inverse XOR-based DPCM, they are written back into a reconstructed UF file, where data will be stored in appropriate locations according to the record structure specified in Figure 1. If data reduction is applied by not transmitting the SN and VV data, then deleted/missing data flags are used as place-holders for these data entries in the reconstructed UF file.

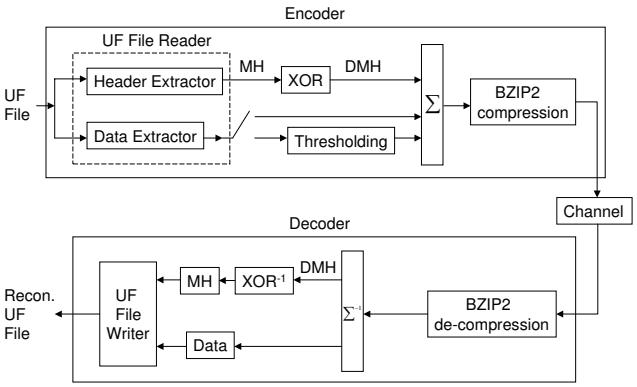


Figure 9: Architecture of UF file encoder and decoder. MH denotes macro-headers, and DMH denotes differential macro-headers. Thresholding is an option at the encoder.

3.4. *Simulation Results*

The results of testing the UF file compression software we developed are summarized in Figures 10 and 11, where it can be seen that data thresholding allowed UF files to be reduced/compressed down to **130 KB** on average, offering more than three times more efficient data reduction and compression than the case without thresholding. Figure 12(a) shows the original reflectivity data of three representative UF files chosen from the test set, with file indices being 1 (00:02:10 UTC), 70 (06:56:38 UTC), and 190 (16:52:17 UTC), respectively. If thresholding was not used, the reconstructed data are near-lossless (except for the small rounding errors discussed in Section 3.2), as can be confirmed both visually such as in Figure 12(b), and numerically (with error guaranteed to be less than 0.5 per data entry). The effect of data blank-out due to thresholding can be observed in Figure 12(c).

Tests also demonstrated that the UF file compression software could meet the near-real-time requirement. Running on a Linux box at NRL, the software took only 3 seconds to compress an UF file, and decompression was even faster – just 1 second was needed.

Figure 10: Sizes of the compressed UF files without using data thresholding. Data reduction was used by keeping three types of data (DZ, VE, and SW). The average size of the compressed files is 444 KB. Thus the size of the original UF files (5.4 MB) was reduced and compressed by over 11 times.

4. SUMMARY AND FURTHER WORK

By efficiently exploiting the inherent redundancies existing in inter-record headers, we were able to losslessly compress headers in UF files to negligibly small sizes. We also introduced data thresholding to achieve larger reduction and compression on radar data, while maintaining sufficient information content to impact analyses and forecasts with the numerical models. We developed a software package for compressing and decompressing radar data files in UF format based on the proposed techniques and an open-source implementation of the widely used *bzip2* algorithms. Depending on the amount of precipitation echo observed throughout SPS-48E radar area coverage, and on the amount of thresholding employed, we were able to compress the UF files down to between 50 KB and 250 KB corresponding to data reduction and compression factor of approximately 40 to 1. Consequently, we were able to achieve more than 3 times more efficient data reduction and compression than what was reported in Kruger and Krajewski (1997).

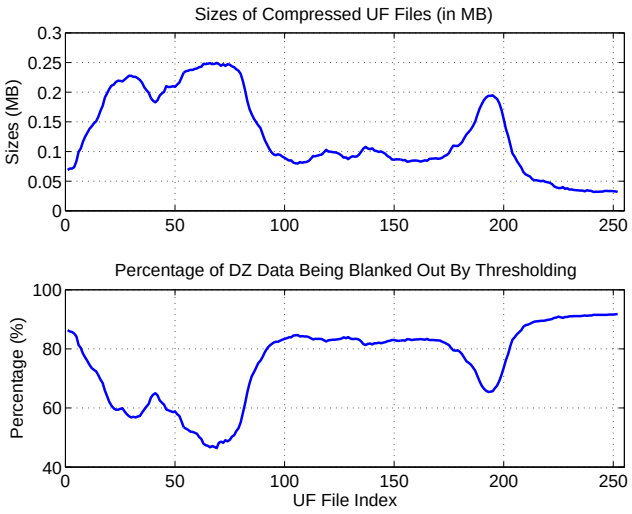


Figure 11: Sizes of the compressed UF files and the associated percentage of DZ data being blanked out by thresholding. The more data being blanked out due to thresholding, the larger the compression achieved on UF files. On average, the size of the compressed files is 130 KB. Thus the size of the original UF files (5.4 MB) was reduced and compressed by over 40 times.

We also estimated a maximum, compressed UF file size of 500 KB for the worst case scenario of more widespread precipitation than what was observed in the current study. This was estimated from the largest compressed UF file containing data that was not thresholded (e.g., data as shown in panel (b) of Figure 12). Therefore, our achieved range of compression ratios would allow UF files of substantially larger size (approximately 10 MB) expected from other Navy radars in the future (e.g., SPS-48G and SPY-1) to be compressed down to below or near the current required 1 MB size for transmission off the ship to a shore location. To further confirm the sufficiency of this requirement, simulations of the transmission of compressed SPS-48E UF files off the ships will also be conducted on ground at the Integrated Test Facility laboratory at SPAWAR. Once confirmed, the developed software package will then be approved for use on ships equipped with the HWDDC. The first at-sea, SPS-48E UF data transmission experiments to FNMOC, along with concurrent UF data assimilation and forecast experiments with COAMPS®, are anticipated in the very near future. Future work is also being considered to investigate additional techniques to further increase the UF file compression ratio.

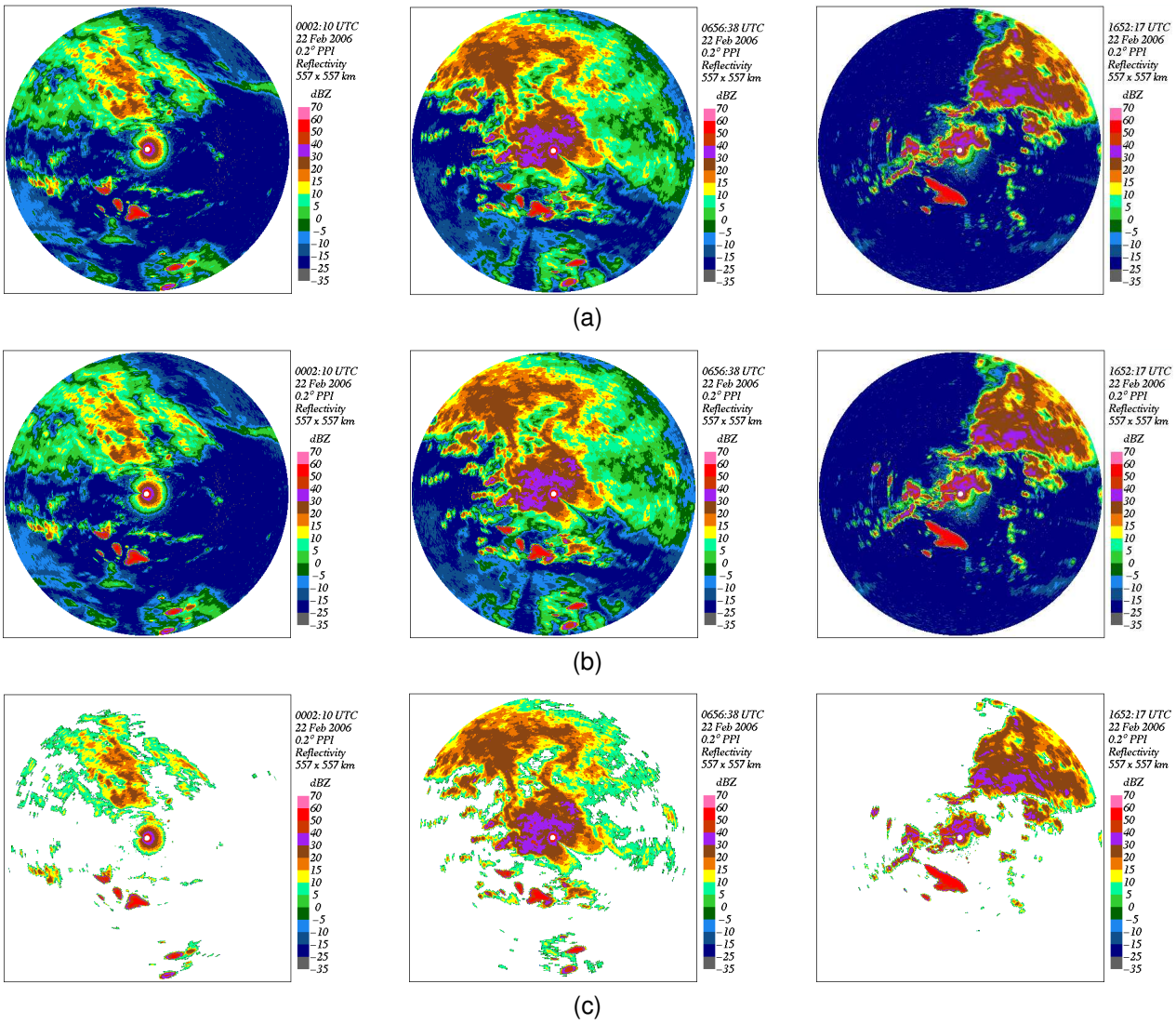


Figure 12: Reflectivity (DZ) data at the lowest tilt at three time points (00:02:10, 06:56:38, and 16:52:17 UTC) on February 22, 2006. (a) Original data. (b) Reconstructed data by using the developed UF file compression software (without data thresholding). (c) Reconstructed data with data thresholding (data entries at locations with either DZ $\leq$ 5 dBZ or SN $\leq$ 10 dB will be set to an identical blank-out marker with value of -100).

Acknowledgement

The first author gratefully acknowledges the support from the Office of Naval Research (ONR) through its Summer Faculty Research Fellowship program, which allowed him to collaborate with other co-authors and conduct this research at the Naval Research Lab, Marine Meteorology Division, Monterey, CA, as a faculty fellow in the summer of 2008.

References

Barnes, S. L., 1980: Report on a meeting to establish a common doppler radar data exchange format. *Bulletin American Meteorological Society*, **61**, 1401–1404.

Cook, J., M. Frost, G. Love, L. Phegley, Q. Zhao, D. A. Geiszler, J. Kent, S. Potts, D. Martinez, T. J. Neu, D. Dismachek, and L. N. McDermid, 2007: The U.S. Navy's on-demand, coupled, mesoscale data assimilation and prediction system. *Proceedings of the 22nd Conference on Weather Analysis and Forecasting/18th Conference on Numerical Weather Prediction*, American Meteorological Society, Park City, UT.

Harasti, P. R., J. Cook, Q. Zhao, L. J. Wagner, C. Kessinger, D. Megenhardt, J. Pinto, and B. Hendrickson, 2006: Extracting weather information from the SPS-48E for US Navy NOWCAST. *Proceedings of the Fourth European Conference on Radar in Meteorology and Hydrology*, Barcelona, Spain, 314–317.

Harasti, P. R., M. Frost, Q. Zhao, J. Cook, L. J. Wagner, T. Maese, S. Potts, J. Pinto, D. Megenhardt, B. Hendrickson, and C. Kessinger, 2007: At-sea demonstration of the SPS-48E radar weather extraction capability. *Proceedings of the 33rd Conference on Radar Meteorology*, American Meteorological Society, Cairns, Australia.

Kruger, A. and W. F. Krajewski, 1997: Efficient storage of weather radar data. *Software Practice and Experience*, **27**, 623–635.

Maese, T., J. A. Hunziker, H. S. Owen, M. Harven, L. J. Wagner, R. Wilcox, G. Cavalieri, and K. Koehler, 2007: Hazardous weather detection and display capability for US Navy ships. *Proceedings of the 23rd AMS Conference on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology*, American Meteorological Society, San Antonio, TX.

Makkapati, V. V. and P. R. Mahapatra, 2007: Extreme compression of weather radar data. *IEEE Transactions on Geoscience and Remote Sensing*, **45**, 3773–3783.

Zhao, Q., J. Cook, Q. Xu, and P. R. Harasti, 2005: Improving very-short-term storm predictions by assimilating radar and satellite data into a mesoscale NWP model. *Proceedings of the 32nd Conference on Radar Meteorology/11th Conference on Mesoscale Processes*, American Meteorological Society, Albuquerque, NM.