Koninklijk Nederlands
Meteorologisch Instituut
Ministerie van Infrastructuur en Milieu

FLYSAFE-2
Final report

M. de Graaf and H. Leijnse

December 24, 2013

UNIVERSITY OF AMSTERDAM

1 Introduction

The FlySafe2 project aims at reducing the risk of bird-aircraft collisions, but also providing the data and models to study fundamental questions about bird behavior along entire migratory flyways. Project partners include the Royal Netherlands Meteorological Institute (KNMI) and the Research Group of Computational Geo-Ecology (IBED-CGE) at the University of Amsterdam (UvA). FlySafe2 is a follow-up of the FlySafe and Bambas projects, and is financed by the Royal Netherlands Air Force (RNLAF).

FlySafe2 builds on the algorithm developed by *Dokter et al.* (2011) in the framework of the FlySafe project for the extraction of profiles of bird densities from radar volume data. Data from radars from several European countries were processed during the FlySafe2 project. The goal was to retrieve profiles of bird densities from data from weather radars across Europe using this algorithm, and to find to what extent this process can be automated. The main challenges for this are:

- The algorithm requires a clutter map that depends on the immediate surroundings of the radar. This map needs to be produced for each radar.
- Each country has its own specific data format. This requires different routines for reading data for each country.

Most meteorological institutes in Europe are members of the EUMETNET Operational Programme for the Exchange of Weather Radar Information (EUMETNET OPERA; see `http://www.eumetnet.eu/opera`). An algorithm that is capable of automatically generating bird density profiles for different types of radars that are operated in varying settings has the potential to be operationally implemented at central radar processing facilities. The OPERA Data Center (ODC or Odyssey) is such a facility that could potentially host this algorithm in the future.

This report describes the work carried out in order to overcome these challenges, and to process data from several radars across Europe. For details regarding the actual bird density retrieval algorithm, the reader is referred to *Dokter et al.* (2011). For more information on operational bird density retrievals and their use in military aviation warnings, see `http://www.flysafe-birdtam.eu/`. The FlySafe2 project was carried out in cooperation with the new European Network for the Surveillance of Animal Movement (ENRAM).

2 Data

Data from radars across Europe were obtained by sending a request for volume data for a test period between 15 August and 15 September 2011 to all OPERA delegates. The OPERA network of operational weather radars is shown in Figure 2.1. Sixteen European countries responded positively to the request to deliver data from their radar systems for the test period (indicated by brighter colors in Fig. 2.1). Some characteristics of the delivered and processed data are listed in table 2.1. Note that not all delivered data were processed. The Belgian data were delivered through a different system and still need to be investigated. The German data were delivered in BUFR format, but a

Figure 2.1. Radars (red dots) available within the OPERA network (green colored countries). The bright countries delivered data for the FlySafe2 project described in this report.

ISO-3166	Country	# radars	original format	remarks
CZ	Czech Republic	2	ODIM HDF5	
FI	Finland	8	ODIM HDF5	
FR	France	17	raw	
HR	Croatia	1	ODIM HDF5	
IE	Ireland	2	ODIM HDF5	
NL	Netherlands	2	raw	
NO	Norway	8	raw	
PL	Poland	4	raw	
PT	Portugal	2	raw	
SE	Sweden	12	ODIM HDF5	
SI	Slovenia	1	ODIM HDF5	
SK	Slovakia	2	ODIM HDF5	
UK	United Kingdom	1	ODIM HDF5	
BE	Belgium	1	ODIM HDF5	Not processed
CH	Switzerland	2	gif	Unknown format
GE	Germany	4	BUFR	Missing tables

Table 2.1. Characteristics of the radar data used in this study. The number of radars in column three is the number of radar systems in that country for which actual data were delivered for the test period. The data formats in column four are: ODIM HDF5 = HDF5 representation of the OPERA Data Information Model, see next section; raw = native radar data format, depending on radar system. BUFR = Binary Universal Form for the Representation of meteorological data; gif = Graphics Interchange Format.

conversion table that is needed to read or convert the data is missing. The Swiss data have an unknown format.

3 Software development

The work performed can be divided into 4 parts:

1. The creation of input-output (I/O) routines for the various data formats from the different countries. This was combined with the conversion of some of the data formats to one common format model, described below.

2. The change of the original bird profile retrieval algorithm, from one stand-alone algorithm written in C language, to a more versatile callable library system, that can be used in any programming language. The library is based on the original algorithm and written in C, to maintain the highest performance in terms of processing speed.

3. The creation of clutter maps for each radar, based on the statistics of the delivered test data (of one month).

4. The creation of the bird profiles, using the I/O-routines, the bird profile library and the generated clutter maps mentioned above, for all data in the test month and for all radars.

3.1. Input/Output

For each country separate I/O-routines were written, because all data from different countries have specific formats and characteristics that must be treated individually, even if the data were converted to a common standard. The standard adopted for this study was the HDF5 representation of the OPERA Data Information Model (ODIM), which is the common format that is being developed for data exchange within the OPERA radar network. More information on ODIM can be found in *Michelson et al.* (2011), available on the OPERA website `http://www.eumetnet.eu/opera`. All radars from which data are available for FlySafe2 are part of the OPERA network. It is envisaged that all OPERA radars will provide data in the ODIM format in the near future.

All data that were not available in ODIM format were converted to ODIM HDF5 format. Although this model was designed to harmonize the data from different systems within Europe and facilitate the collaboration between countries and the exchange of data, it still leaves enough room for differences between various data systems, that must be treated individually. For example, multiple radar data products are collected for various scans (heights). Within the ODIM model these various products and scans can be collected within one (HDF5) file, or the scans can be divided over different files,

or the radar products can be divided over different files. All combinations are possible and all combinations exist within the delivered test data.

Therefore, country-dependent I/O-routines were created to manage these differences, while the bird profile retrieval algorithm is the same for all systems. The I/O-routines were written in IDL language, for convenience of the programmer, and to facilitate the visualization of the end products. However, the programming language of the I/O-routines is not restricted, and can be written using any language.

The communication of the I/O-routines with the callable C bird retrieval routines is through (C) structures, containing the DBZH scans, the VRAD scans and the clutter map data for each scan, and their meta data, also collected in structures. The retrieval algorithm returns the cell map and VVP velocity texture in structures and the profile quantities in arrays. In the current set-up these are collected in the IDL I/O-routines and written to disk for visualization and storage, in a format similar to ODIM HDF5 with some added features.

3.2. Bird retrieval algorithm

The original bird retrieval algorithm was available as a standalone C-program, using native KNMI HDF5 radar files for input. The output was a HDF5 file containing the bird density profile for that radar, optionally with the original scans included in the output files. The various options were included in the C-program, and could be changed manually using command-line switches. It is described in detail in *Dokter et al.* (2011).

The algorithm was changed in several ways. The most important change was the separation of the I/O-part of the algorithm from the computation core. Because C-language provides the best performance in terms of speed, the computation core was retained in C-language. All functions and procedures were separated from the data stream and contained in a library that can be called from within any programming language. The communication is through C-structures, defined in the technical guide. In this configuration any specific data format can be handled in separate routines, as long as the data is presented in a structure that is recognized by C. This can be in a C-routine, in IDL, as in the current study, or in any other compatible programming language, like FORTRAN or PYTHON.

Furthermore, since different radar systems have different characteristics, the various options for computing the bird density from different systems must be different. This can now also be defined in the system dependent I/O routines. The same bird retrieval library can then be called with different options for different radar systems.

Lastly, the output was changed to match the ODIM specifications, which is different from the native KNMI HDF5 format that was used before. There is no specification for (bird density) height profiles in the ODIM model, but these were added as separate arrays within the HDF5 file, following the specification in *Dokter et al.* (2011). This can easily be recognized using the HDF5 format.

The technical description of the bird retrieval algorithm in IDL is given in Appendix A.

3.3. Clutter maps

Most radars for which data were available employ Doppler clutter filters. Such filters remove reflectivities that originate from stationary targets. This effectively removes

most ground clutter. However, the quality of the data after filtering may be severely impacted. Especially of the return from a stationary object is very strong, any other signal may be masked by this, even after filtering. This is why pixels where the clutter filter is observed to remove a very large fraction of the total received power are set to zero in an operational setting. The problem with such pixels is that there is generally no information available about whether this pixel is zero because there was no return, or because its quality was judged to be too low. The use of the bird density retrieval on such data will cause underestimation of bird densities. There are two ways to deal with this. The first is to determine which pixels have been significantly affected by a clutter filter by comparing corrected reflectivities (DBZH in ODIM terms) to uncorrected reflectivities, and to remove these pixels from further analyses. The second is to make clutter maps based on uncorrected reflectivity data to determine which pixels are affected by clutter, which can then be removed from further analyses.

Another problem with clutter is that some of it may not be filtered because either no clutter filter is implemented at the radar, or the clutter is not stationary (e.g. trees and other vegetation). This will cause residual clutter returns to be present in the data. If the bird density retrieval algorithm is implemented on radar data without correcting for this, an overestimation of bird densities will occur.

It is therefore important to know where (in which pixels) clutter is present in the radar data employed for bird density retrievals. The pixels that are known to be affected by clutter can then be excluded from further analyses. *Dokter et al.* (2011) used several days of dry and insect- and bird-free data to construct a clutter map, which was then used to exclude all pixels exceeding -10 dBZ. This clutter map was simply the average of all uncorrected reflectivities (i.e. no clutter filter was applied; TH in ODIM terms) over that period. This proved to be very effective. However, it is not feasible to do this for a large number of radars, as this would require selecting suitable periods for building clutter maps by hand for each radar. Another issue with a clutter map generated for such a period, is that the surroundings of the radar may change over time (e.g. new buildings are constructed, trees grow).

A new method to automatically derive clutter maps was therefore developed. This method is based on the statistics of a given radar pixel over a given period. If this period is long enough (say, one month), it can be assumed that there is no rain, insects or birds for at least a given fraction of the period (say, 25 percent). The 25^{th} percentile of the distribution of the values of uncorrected reflectivity (TH) for a given pixel will then be a good indicator of the level of clutter present at that pixel. Taking the 25^{th} percentile for each pixel will then yield a clutter map that can be used in the bird density retrieval algorithm.

To test whether the automatically generated clutter maps are realistic, and to determine the quantile that should be used for generating clutter maps, the manually generated clutter maps by *Dokter et al.* (2011) were used. The focus here is on the two Dutch radars (De Bilt and Den Helder) because of the availability of these clutter maps and a long archive of data. For each pixel within 25 km from the radar, the percentile in the distribution of uncorrected reflectivity that corresponds to the value from the manually generated clutter map is recorded, resulting in a percentile value per pixel. The distribution of these percentile values can then be analyzed to give insight into how well an automatically generated clutter map would capture the true clutter, and what percentile should be used for generating such clutter maps. This can be done for different months to gain insight into the stability of the automatically generated clutter maps. Figure 3.1 shows these distributions computed over 2011.

It is clear from these figures that there is certainly not a single percentile value that

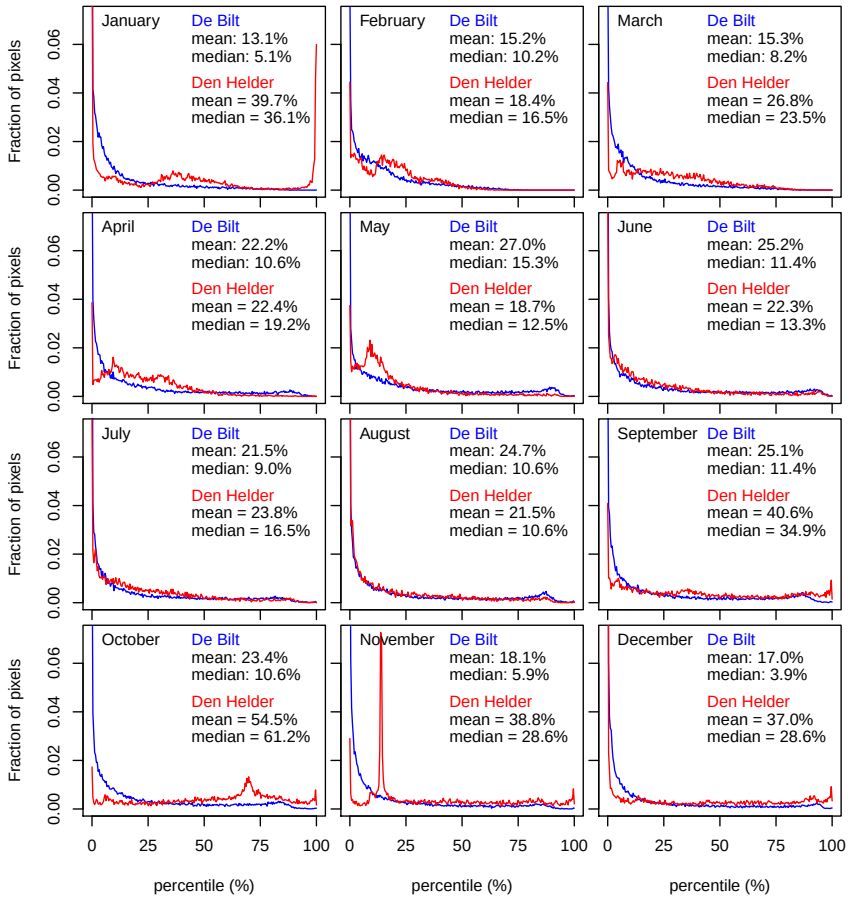

Figure 3.1. Distribution of the percentiles of the distribution of uncorrected reflectivity over a given month corresponding to values in the manually generated clutter maps, for radars in De Bilt (blue) and Den Helder (red), for 2011.

can be used to perfectly reproduce the manually generated clutter maps. This does not mean that this method cannot be used to remove pixels that are affected by clutter from further analysis. The percentile distributions for the De Bilt radar are seen to be relatively constant over a year. This is not the case for the Den Helder radar. The explanation for this is that this radar is very close to the sea, and most pixels within 25 km from the radar are over sea. Sea clutter is known to be highly dependent on atmospheric propagation conditions, which are known to vary throughout the year. This results in significantly different distributions of percentiles for each month. This is shown in Fig. 3.2, where the medians of the distributions of percentiles (Fig. 3.1) are plotted as a function of the month of 2011.

Figures 3.1 and 3.2 show that more research is needed to further assess the quality of automatically generated clutter maps. The importance of applying such clutter maps will vary greatly with radar location. The combination of the employed scan schedule and the area immediately surrounding the radar determines the extent and severity of

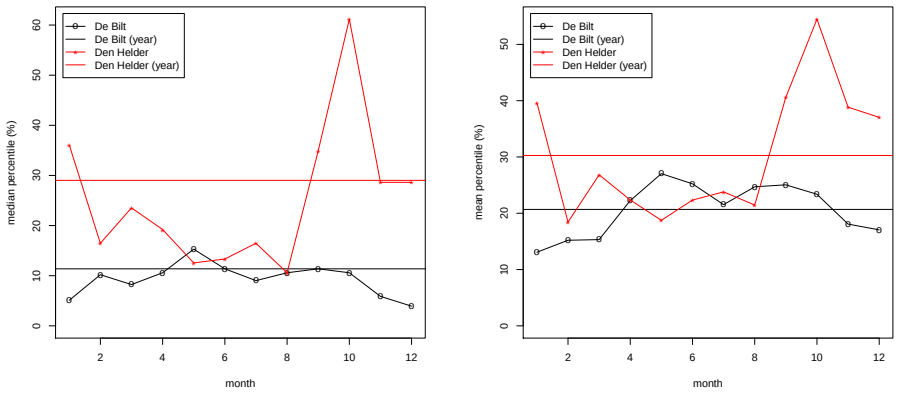

Figure 3.2. Medians (left) and means (right) of the distributions of percentiles of the distribution of uncorrected reflectivity over a given month (or the entire year) corresponding to values in the manually generated clutter maps, for radars in De Bilt (blue) and Den Helder (red), for 2011.

the clutter. The fact that both may change (change in operational scan schedule, and buildings being constructed or trees growing) stresses the need for an algorithm that can be used to automatically generate clutter maps.

For most of the radars for which data were made available, no uncorrected reflectivity data were included. This means that pixels that were set to zero because the clutter filter removed a very large fraction of the returned power are indistinguishable from those where the returned power was zero. The clutter maps that have been generated are hence based on corrected reflectivity data so that the application of these clutter maps corrects for pixels that often experience residual clutter. Clutter maps were generated for 0^{th}, 10^{th}, 25^{th}, 50^{th}, and 75^{th} percentiles, and the mode of the distribution of values for a given pixel. The clutter maps that are used subsequently in this report are those generated based on the 0^{th} and 25^{th} percentiles, where the 0^{th} percentile clutter map corresponds to not applying a clutter map.

4 Bird density profiles

Bird density profiles were generated from all available data from the 62 radars described in this report, for the entire test period. Like in *Dokter et al.* (2011) the clutter maps were applied by removing all pixels where the values in the clutter maps exceed -10 dBZ. The entire data set was first processed using a zero-percentile clutter map (no extra clutter filtering). An example of the reflectivities used for the processing is shown in Figure 4.1 (left panels). In this figure a composite PPI plot was created using the reflectivity fields of the lowest scan of each radar. Note that the lowest scans do not necessarily refer to the same heights, as the radar heights and the scan angles may vary

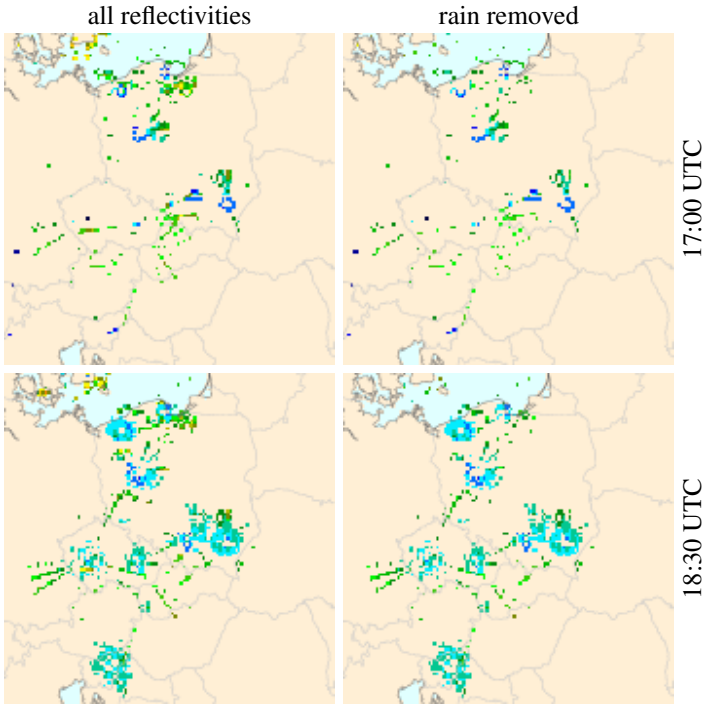

Figure 4.1. PPI composite of reflectivities of the lowest scans of several central European radars on 15 Sep. 2011 at 17:00 UTC (top panels) and 18:30 UTC (bottom panels). Left-hand panels show total reflectivities, and right-hand panels show only those reflectivities that the algorithm classifies as bird echoes.

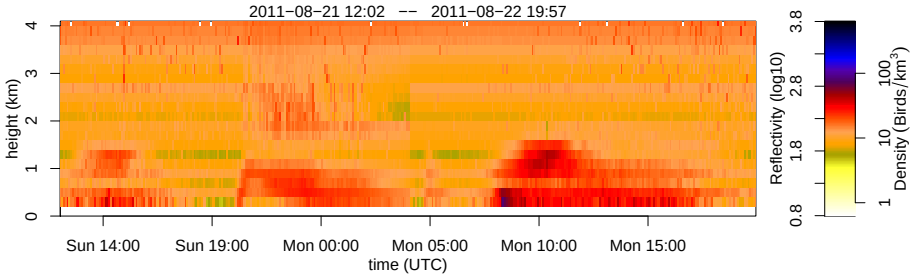

Figure 4.2. Bird density profiles from 0 - 4 km altitude during 21 Aug. 2011 as a function of time, averaged from 5 - 25 km distance of the Chenies radar, UK.

between radar systems. The PPIs are shown for two moments in time: 15 Sep. 2011 at 17:00 UTC and 15 Sep. 2011 at 18:30 UTC.

In the right panels of Fig. 4.1 the same composite PPIs are shown as before, but now the data are filtered for rain and clutter, i.e. all radar cells that were classified as rain or clutter were removed. The effect is clear from a comparison of both figures: most of the reflectivities are attributed to rain and/or clutter and most of the signal is removed. However, from a comparison of the filtered PPIs at 17:00 UTC and 18:30 UTC (cf. top-right and bottom-right panels in Figure 4.1) an intensification of reflectivities can be observed in central Europe, i.e. Poland, Slovakia and Slovenia.

Bird profiles extracted from data from the Chenies radar (UK) for Aug. 21, 2011 are shown in Fig. 4.2. This radar is very close (17 km to the south-west) to the entomological radar in Rothamsted, which makes it very interesting for cross-taxa studies. However, the noise level of the Chenies radar is known to be high. This means that the detectability of bird migration or other low intensity signals is limited. This is clear from Fig. 4.2, where the noise level of this radar corresponds to approximately 7 birds per km^3, so the bird density should be greater than this in order for it to be detectable. However, Figure 4.2 also shows that it is possible to see bird migration on such high-noise radars.

In Figures 4.3-4.5 the bird profiles are plotted for some selected radar systems, showing the development with time of the bird density during this day at 5 - 25 km around these radars and between 0 - 4 km altitude. Most radar systems do not show a clear increase of bird density after 18:00 UTC, except for the radars in the last figure, which are located in Central Europe. This may be an indication of increased bird intensity in this region from about 18:00 UTC.

The clutter maps are available in directory `cluttermaps`. The bird density profiles as described above can be found in `zero_bird`. The bird densities have also been determined for a clutter map definition of 25% and settings for small passerines (see *Dokter et al.* (2011), eq. 2.15). These can be found in directory `bird`.

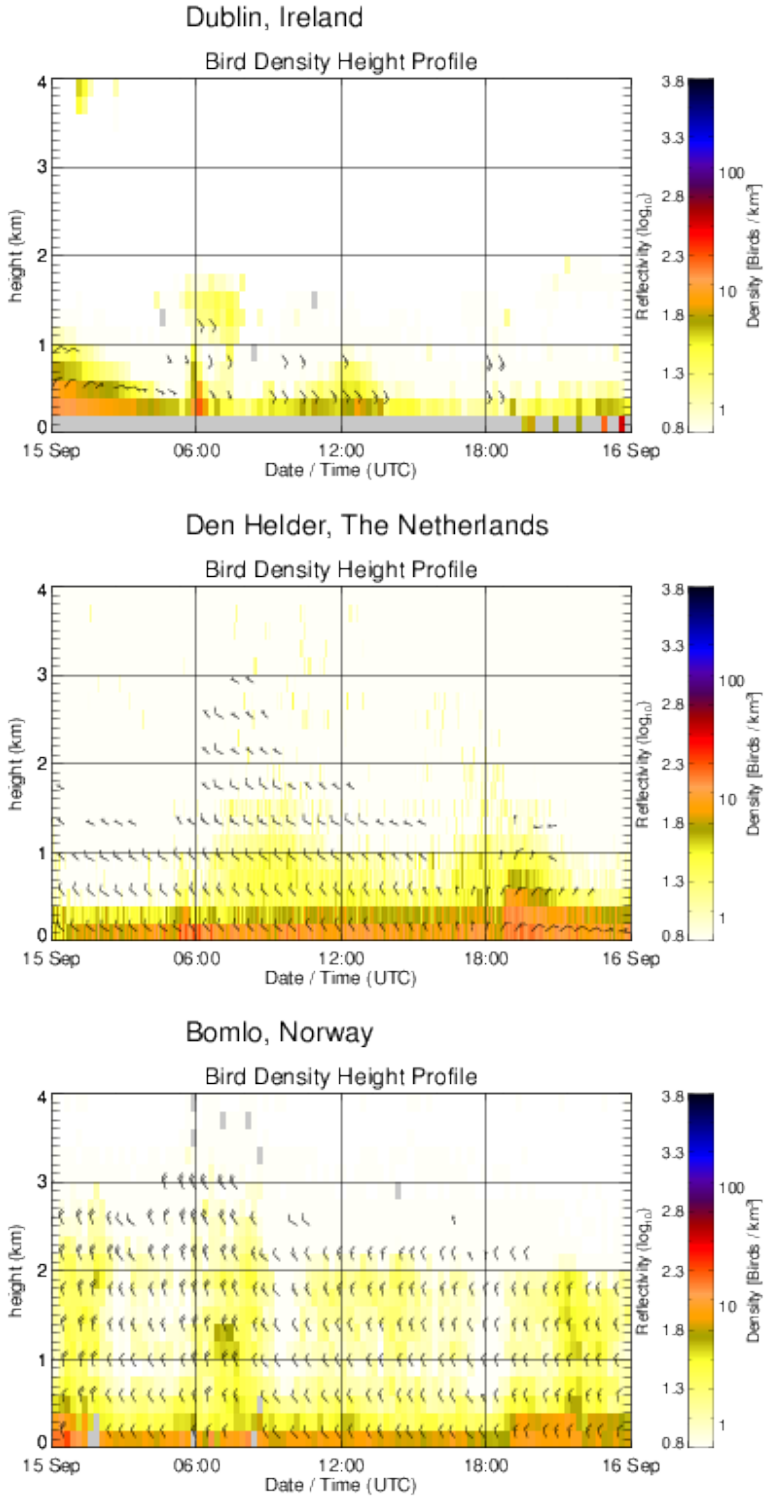

Figure 4.3. Bird density profiles from 0 - 4 km altitude during 15 Sep. 2011 as a function of time, averaged from 5 - 25 km distance of the indicated radar system.

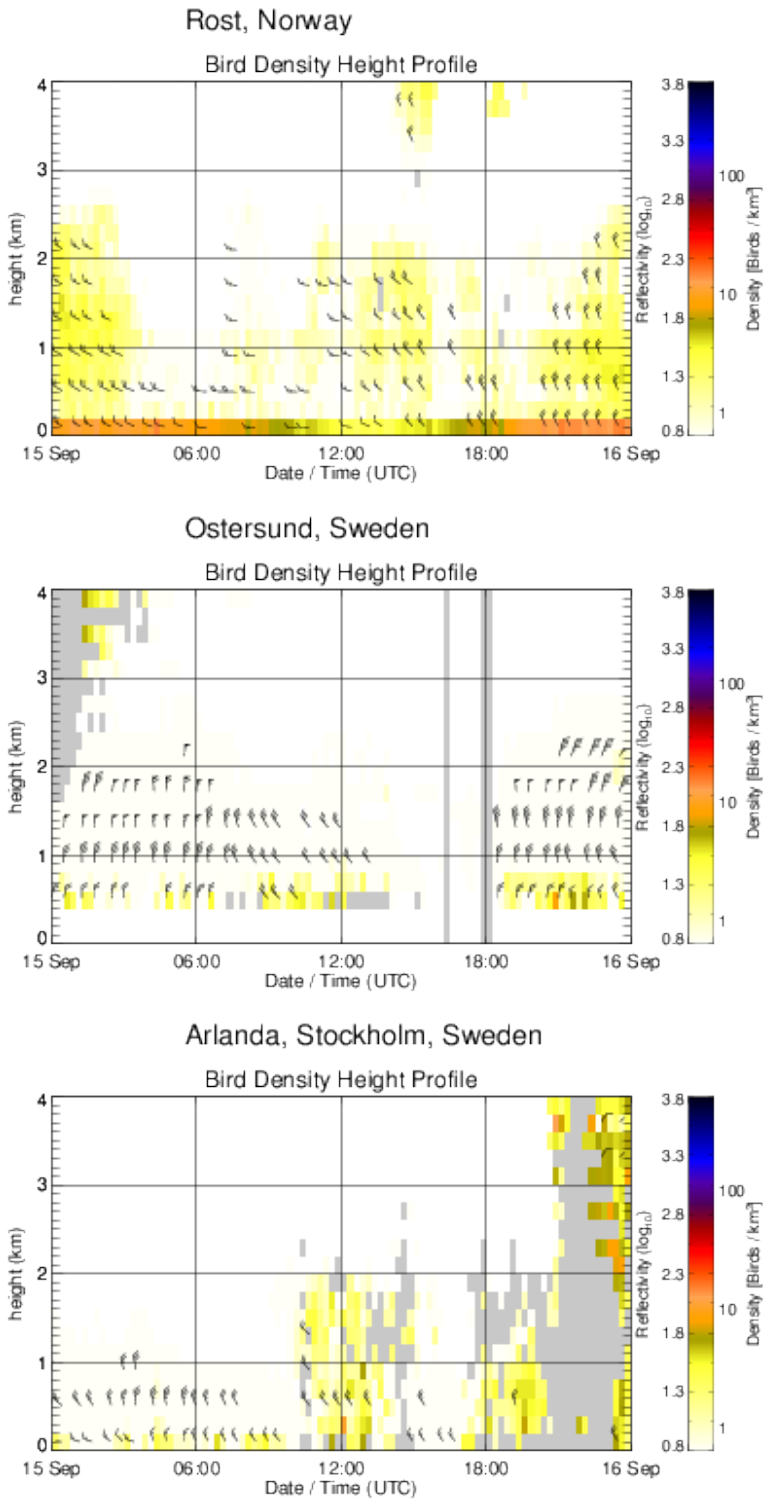

Figure 4.4. Bird density profiles from 0 - 4 km altitude during 15 Sep. 2011 as a function of time, averaged from 5 - 25 km distance of the indicated radar system.

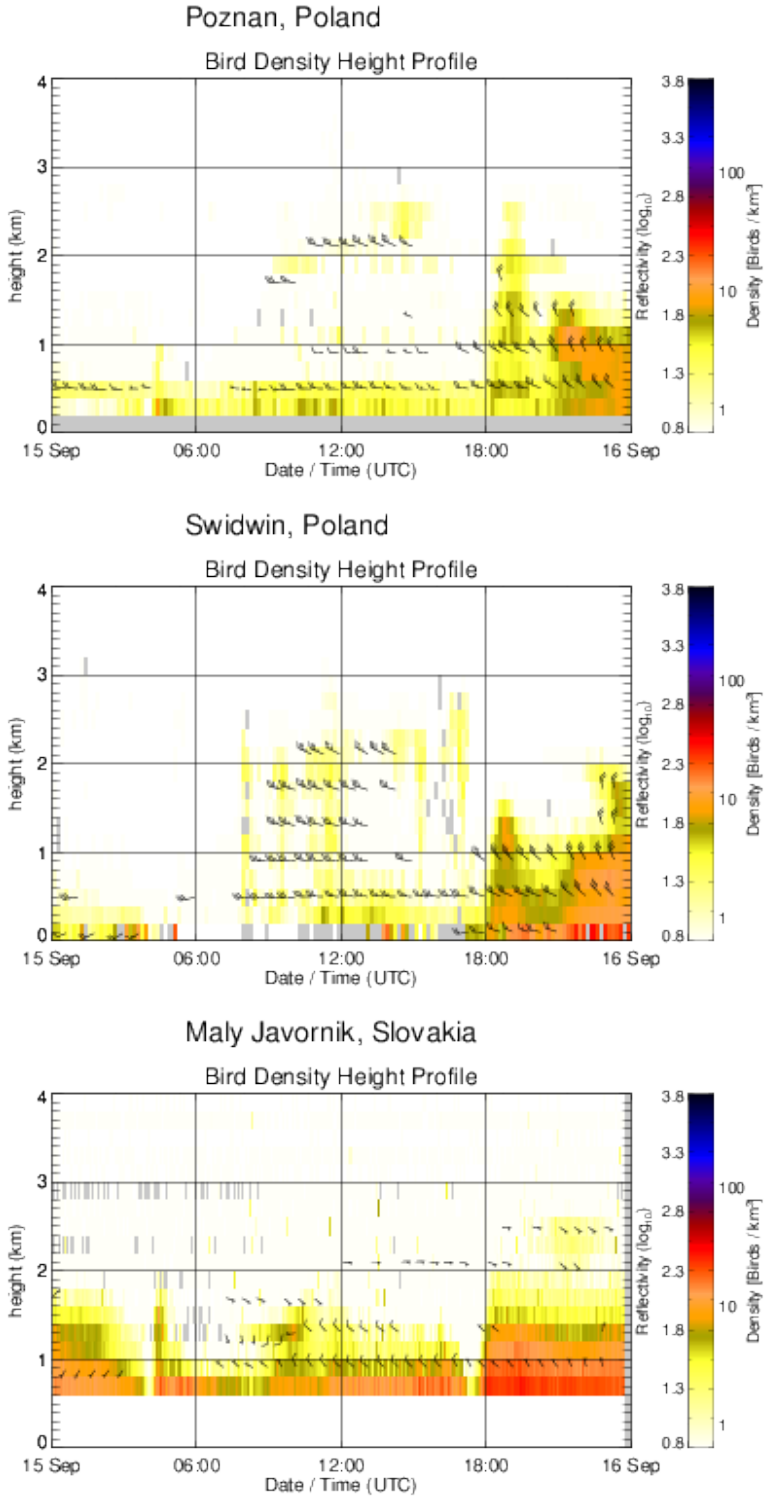

Figure 4.5. Bird density profiles from 0 - 4 km altitude during 15 Sep. 2011 as a function of time, averaged from 5 - 25 km distance of the indicated radar system.

Appendix A

Algorithm Technical description

A.1. IDL port

The IDL port of the algorithm uses C-libraries to do the actual data processing. These libraries are also used by the algorithm written in ANSI C, ensuring identical data processing independent of the used software language, while a high processing speed is ensured by using C. The C-libraries can potentially be called using any programming language, providing flexibility for the programmer. The intention is easy creation of data format conversion tools using high level languages, while keeping the efficiency of C. The cost is a larger effort of the programmer to design cross-language interfacing tools and some overhead for calling external libraries. This document helps the programmer by describing the input/output of the C-libraries in detail.

To date the algorithm is provided in two languages: IDL and C.

The algorithm in IDL is built as follows. The core algorithm is called `bird_call.pro`. This routines calls the C-library `vol2bird.so`, in which the bird density retrieval C-functions are defined. This library can be created from `vol2bird_routines.c`. The routine `bird_call.pro` also calls the IDL I/O-procedures, which provide the interface to the actual data. These I/O-procedures are defined for each country, and are named according to the following format: `vol2bird_cc_dataformat.pro`, where `cc` means *country code* and `data_format` refers to the format of the data that is provided. E.g. the IDL I/O-procedure for data from France is provide by the routine `vol2bird_fr_odimhdf5.pro`, because the country code for France is FR, and the data from France have been converted into ODIM standard. The I/O-procedures return the necessary data for the algorithm in IDL structures, which is directly compatible with C-language, and can be fed to `vol2bird`. Then `bird_call.pro` computes the bird density profiles and calls the I/O-procedure again to write the data in HDF5 in an ODIM-like format. Please note that no definitions exist for the bird profiles within ODIM. The profiles are added in the HDF5 file in an easily recognizable format. The available I/O-procedures are listed in table A.1. Note that two procedures are available for The Netherlands: one for the data converted into ODIM-format, and one for the native KNMI HDF5 format that was used in the original C-language bird density algorithm.

Within the I/O-procedures all routines start with the convention `vol2bird_cc_dataformat_command`. E.g. the command for reading a scan

CC	Country	I/O-procedure name
CZ	Czech Republic	`vol2bird_cz_odimhdf5.pro`
FI	Finland	`vol2bird_fi_odimhdf5.pro`
FR	France	`vol2bird_fr_odimhdf5.pro`
HR	Croatia	`vol2bird_hr_odimhdf5.pro`
IE	Ireland	`vol2bird_ie_odimhdf5.pro`
NL	Netherlands	`vol2bird_knmihdf5.pro`
NL	Netherlands	`vol2bird_nl_odimhdf5.pro`
NO	Norway	`vol2bird_no_odimhdf5.pro`
PL	Poland	`vol2bird_pl_odimhdf5.pro`
SE	Sweden	`vol2bird_se_odimhdf5.pro`
SI	Slovenia	`vol2bird_si_odimhdf5.pro`
SK	Slovakia	`vol2bird_sk_odimhdf5.pro`

Table A.1. Available I/O-procedures

from Irish data is `vol2bird_ie_odimhdf5_read_scan`, which is used within `bird_call.pro` to call the correct routine for reading the data. This makes it possible to dynamically switch from one radar system to another within the bird retrieval algorithm.

For each radar station that is called, a description must be provided of the I/O-routines that should be used. This is provided in the `radar_definitions.pro` routine. In this file all radars are uniquely defined within IDL, and some useful information is described. These are returned to the calling program within a structure names `radar_definition`. E.g. the Swedish radar system at Ångelholm has the following information in `radar_definition`.

```
** Structure <9345a8>, 9 tags, length=144, data length=144, refs=1:
   RADAR_FULL_NAME STRING    'Sweden_Angelholm'
   RADAR_ID        STRING    'seang'
   PATH_ID         STRING    'SE_ang'
   IO_FILE         STRING    '~/FLYSAFE/idl/io/vol2bird_se_odimhdf5.pro'
   RAW_DATA_PATH   STRING    '~/FLYSAFE/process/data/raw/'
   INPUT_DATA_PATH STRING    '~/FLYSAFE/process/data/odim/'
   BIRD_DATA_PATH  STRING    '~/FLYSAFE/process/data/bird/'
   CLUTTER_DATA_PATH STRING  '~/FLYSAFE/process/data/cluttermaps/'
   ASCII_NAME      BYTE      [142,110,103,101,108,104,111,108,109]
```

`RADAR_FULL_NAME` is a string that gives the human readable name of the system, with the name of the country and the radar name separated by an underscore _. `RADAR_ID` is a string that uniquely identifies a data file of that system (so this string should be part of any (and only this) data file from this system). `PATH_ID` is a 6 element string identifying this system within the data structures of this project. It is always constructed from the two element country code is capital letters, an _, and a three element string abbreviations of the actual name. `IO_FILE` points to the I/O-procedure file to be used for this system. `RAW_DATA_PATH`, `INPUT_DATA_PATH`, `BIRD_DATA_PATH`, and `CLUTTER_DATA_PATH` refer to the directories containing the raw data, prepared input data, output data, and clutter maps for this system, respectively. `ASCII_NAME` is an option field that contains the name of the system in extended ASCII characters, entered using the ASCII code in BYTES.

All radar systems are identified using a unique 5 element string, that is the same as the PATH_ID string without the _. RADAR_NAMES.PRO provides an easy interface for getting the radar id's of all (62) radar systems, or from one or a few countries. Using the radar id's, the algorithm can easily be run for many systems in a row.

A.2. Internal communication

When the data have been read, it is collected in a structure with all data in BYTE arrays called scan*n*, that can have different sizes. The first element of the data gives the total number of scans. An example of the radial velocity data VRAD, read for twelve scans is given below:

```
** Structure <1f21d08>, 13 tags, length=1326604, data length=1326604,
refs=1:
   VSCAN           LONG               12
   SCAN1           BYTE      Array[500, 360]
   SCAN2           BYTE      Array[500, 360]
   SCAN3           BYTE      Array[500, 360]
   SCAN4           BYTE      Array[500, 360]
   SCAN5           BYTE      Array[367, 360]
   SCAN6           BYTE      Array[205, 360]
   SCAN7           BYTE      Array[500, 360]
   SCAN8           BYTE      Array[200, 360]
   SCAN9           BYTE      Array[168, 360]
   SCAN10          BYTE      Array[124, 360]
   SCAN11          BYTE      Array[76, 360]
   SCAN12          BYTE      Array[45, 360]
```

The meta data is collected into an array of structures. An example of a *vmeta* IDL structure array is given below:

```
VMETA           STRUCT    = -> <Anonymous> Array[13]
```

The number of dimensions of the meta structure to the number of scans+1, because the first structure (vmeta[*0*]) is left blank, and vmeta[*n*] is passed with the data corresponding to the data of scan *n*

```
** Structure <1eb7328>, 20 tags, length=80, data length=80, refs=2:
   DATE            LONG          20110902
   TIME            LONG              110
   HEIG            FLOAT          0.139000
   ELEV            FLOAT           2.99927
   NRANG           LONG               500
   NAZIM           LONG               360
   RSCALE          FLOAT          0.500000
   ASCALE          FLOAT           1.00000
   AZIM0           LONG                 0
   ZOFFSET         FLOAT          -327.680
   ZSCALE          FLOAT           2.56000
   MISSING         LONG                 0
   PRFH            FLOAT           0.00000
   PRFL            FLOAT           0.00000
   PULSE           FLOAT           0.00000
```

```
   RADCNST          FLOAT           0.00000
   TXNOM            FLOAT           0.00000
   ANTVEL           FLOAT           0.00000
   LAT              FLOAT           60.9036
   LON              FLOAT           27.1111
```

The data and meta data are passed back to the IDL calling routine and passed to the C-library. Within the C-library the data and meta data are received using arrays and structures, that must match the offered data structures exactly. The meta data structure definition in C is given below:

```
*struct scanmeta {
   int date;       /*Date of scan data in YYYYMMDD.*/
   int time;       /*Time of scan data in HHMMSS.*/
   float heig;     /*Height of radar antenna in km.*/
   float elev;     /*Elevation of scan in deg.*/
   int nrang;      /*Number of range bins in scan.*/
   int nazim;      /*Number of azimuth rays in scan.*/
   float rscale;   /*Size of range bins in scan in km.*/
   float ascale;   /*Size of azimuth steps in scan in deg.*/
   int azim0;      /*Ray number with which radar scan started.*/
   float zoffset; /*Offset value of quantity contained by scan.*/
   float zscale;   /*Scale of value of quantity contained by scan.*/
   int missing;    /*Missing value of quantity contained by scan.*/
   float PRFh;     /*High PRF used for scan in Hz.*/
   float PRFl;     /*Low PRF used for scan in Hz.*/
   float pulse;    /*Pulse length in microsec.*/
   float radcnst; /*Radar constant in dB.*/
   float txnom;    /*Nominal maximum TX power in kW.*/
   float antvel;   /*Antenna velocity in deg/s.*/
   float lat;      /*Latitude of the radar
   float lon;      /*Longitude of the radar
};
```

Note that the definition of an INTEGER in C corresponds with a LONG in IDL. Using these definitions the C-library can be called and the data can be passed back and forth.

A.3. output

The output of the algorithm is written into an HDF5 file, in the directory defined by `BIRD_DATA_PATH` and `RADAR_ID`. The filename has the format RAD`_PATH_ID`PRF*_datetime*.h5, where *datetime* is the file's date and time in UTC. The file contains the input data (the logged horizontally-polarized total uncorrected reflectivity factor (TH), if available, the logged horizontally-polarized total corrected reflectivity factor (DBZH), and the radial velocity (VRAD) in ODIM format), and two derived horizontal fields: CELLMAP, a mask containing the identified rain cells, and VTEX, the wind velocity texture field, also in ODIM format. Additionally, the derived bird, precipitation (non-bird) and wind profiles are written into the output file. No ODIM definitions exist for these vertical one-dimension arrays, but they are self-contained and easily recognizable in the HDF5 file. The format defined in Appendix B of *Dokter et al.* (2011) is followed: `profile1` contains the bird profiles, `profile2` contains the non-birds profiles, and `profile3` contains the wind profiles.

References

Dokter, M., Adriaan, Felix Liechti and Iwan Holleman, Bird detection by operational weather radar, *KNMI scientific report; WR 2009-06*, 2009.

Daniel B. Michelson, Rafał Lewandowski, Maciej Szewczykowski, and Hans Beekhuis, EUMETNET OPERA weather radar information model for implementation with the HDF5 file format, *OPERA Working Document WD_2008_03*, 2011.