Koninklijk Nederlands
Meteorologisch Instituut
Ministerie van Infrastructuur en Milieu

FLYSAFE-2
Final report

M. de Graaf

September 9, 2013

UNIVERSITY OF AMSTERDAM

1 Introduction

The flysafe2 project aims at reducing the risk of bird-aircraft collisions, but also provide the data and models to study fundamental questions about bird behavior along entire migratory flyways. Project partners include the Royal Netherlands Institute for Meteorology (KNMI) and the Research Group of Computational Geo-Ecology (IBED-CGE) at the University of Amsterdam (UvA). FlySafe2 is a follow-up of the Flysafe1 and Bambas projects, and is financed by the Royal Netherlands Airforce.

In this Flysafe2 final report the process of extracting bird densities from European (weather) Doppler radars is descibed. The various processed radars are discussed, and the way the data between the radars were synchronised. The results from this projects should help to forecast of bird migration intensity over The Netherlands and Belgium.

More information can be found on the internet:
`http://www.flysafe-birdtam.eu/`.

The effort is coordinated in close cooperation with the new European Network for the Surveillance of Animal Movement (ENRAM).

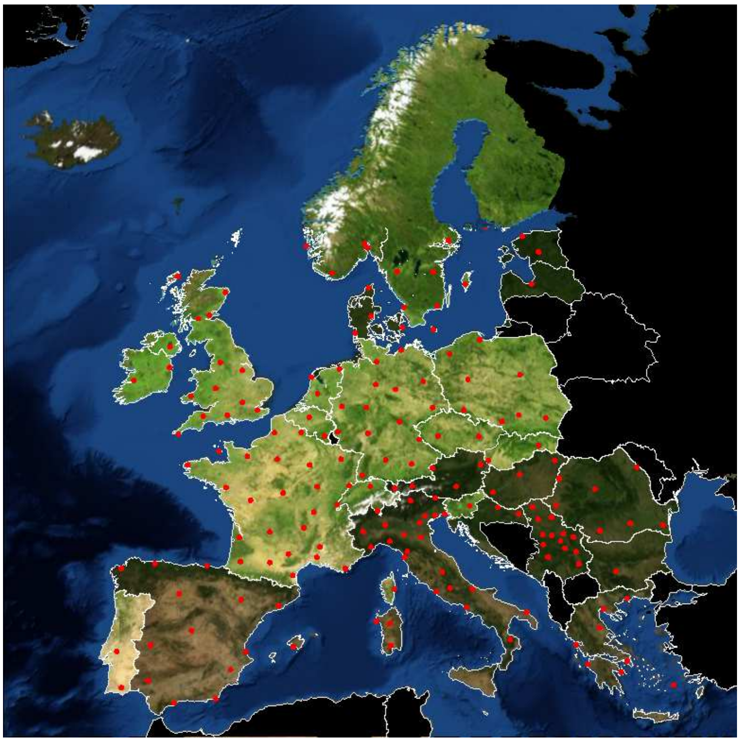


Figure 1.1. Radar systems (red dots) available within the OPERA network (green coloured countries). The bright countries delivered data for the Flysafe-2 project described in this report.

Data from various radars from several European countries were processed during the Flysafe2 project. The goal was to retrieve bird profile densities from the weather radar returns, by means of the Vol2Bird Algorithm (*Dokter et al.*, 2011), and to find to which extend this process can be automated. In order to do so, a request was sent out to send data for each radar system within the OPERA network (see Figure 1.1) during a test period of 15 Aug. 2011 – 15 Sep. 2011. Sixteen European countries responded to the call to deliver data from their radar systems from this period. The sixteen countries are indicated in Figure 1.1. The statistics of the delivered and processed data are listed in table 1.1. Note that not all delivered data was processed (yet). The Belgium and UK data were delivered through a different system and still need to be investigated. The German data were delivered in BUFR format, but a conversion table is missing which is needed to read or convert the data. The Swiss data have an unknown format.

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	hdf5	
NO	Norway	8	rainbow 4	
PL	Poland	4	rainbow 4	
PT	Portugal	2	raw	
SE	Sweden	12	ODIM hdf5	
SI	Slovenia	1	ODIM hdf5	
SK	Slovakia	2	ODIM hdf5	
BE	Belgium	1	ODIM hdf5	Not processed
CH	Switserland	2	gif	unknown format
GE	Germany	4	BUFR	Missing definition tables
UK	United Kingdom	?	?	Not processed

Table 1.1. Statistics of the radar systems used in this study. The number of radars in column three is the number of radar systems in that country for which actual data was delivered for the test period. The data formats in column four are: ODIM hdf5 = common data model for OPERA network, 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 Form.

2 Software development

The work performed can be devided into 4 parts:

1. The creation of input-output (IO) 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 cluttermaps for each radar station, based on the statistics of the delivered test data (of one month).
4. The creation of the bird profiles, using the IO-routines, the bird profile library and the generated cluttermaps mentioned above, for all the data in the test month for all stations.

2.1. Input/Output

For each country separate IO-routines were written, because all data from different countries have specific formats and characteristics that must be treated individually, even if the data was converted to a common standard. The standard adopted for this study was the EUMETSAT OPERA weather radar information model (ODIM) in HDF5 format, which is the common format that is being developed within the European OPERA radar network. This model was defined in the EUMETSAT OPERA weather radar information model for implementation with the HDF5 file format document (*Michelson et al.*, 2011), available on the OPERA website:
`http://www.knmi.nl/opera/opera3/`
All stations are part of the OPERA network and the intention is to implement the ODIM format for all stations within the network in the future, if not done so already.

All the data that were not already in ODIM format were converted to ODIM hdf5 format. Although this model was designed to harmonise 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 devided over different files, or the radar products can be devided over different files. All combinations are possible and all combinations exist within the delivered test data.

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

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

2.2. 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 IO-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 librabry 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 recognised 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.

Furtermore, 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 IO 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 seperate arrays within the HDF5 file, following the specification in *Dokter et al.* (2011). This can easily be recognised using the HDF5 format.

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

2.3. Cluttermaps

Clutter filtering is very important to separate the small bird returns from precipitation and static clutter. Since no cluttermaps were delivered with the test data and clutter can

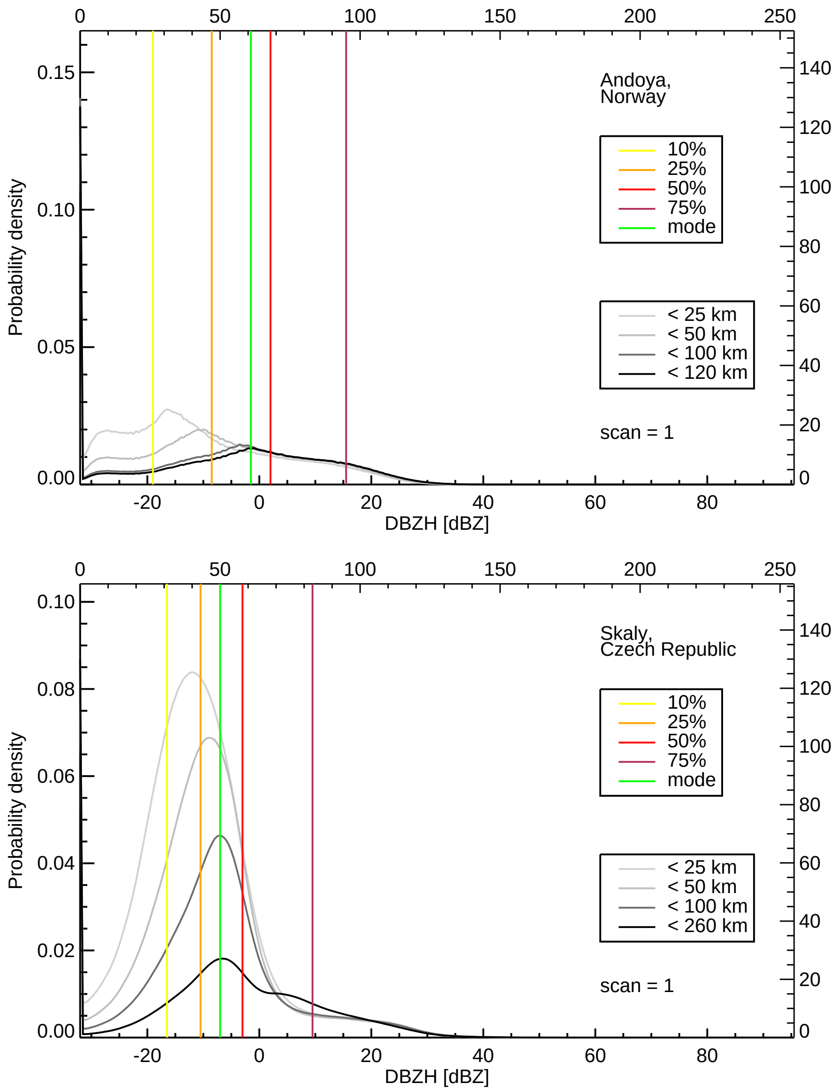


Figure 2.1. Distribution of the radar return signal during 15 August to 15 September for the Norwegian radar system in Andoya (top panel), and the Czech system in Skály (bottom panel), averaged over all radar cell (black lines), all radar cells within 100 km (dark grey line), within 50 km (grey line) and within 25 km (ligh grey line). The percentiles for the distribution of the signals of all cells is given in the lines colored yellow (10%), orange (25%), red (50%) and purple (75%), while the mode is given in green.

be variable, an automated clutter generation routine was developed. This was based on the statistics of the delivered test data, covering one month. For each radar cell the statistics of the radar returns was computed for the test period of 15 August to 15 September. The distribution of the return signal was determined for each cell during this period. A clutter map was then created by defining clutter when a certain percentage of the signal was larger than a threshold value. The threshold value was set at -10 dbZ, and cluttermaps were created for 0, 10, 25, 50, and 75 percentiles, and the mode of the distribution. All pixels with values larger than the threshold value for a greater percentage of the time than the set percentile were treated as clutter and removed from the analysis. Examples of the distribution of the return signal during the test period for the radar system in Andoya, Norway and the Czech system in Skály are given in Figure 2.1. For the cluttermap the distribution and the percentiles were determined for each radar pixel separately.

Currently, the cluttermaps do not give the desired results, and bird density profiles have also been determined for each radar system using a cluttermap defined by the zero percentile (directory `zero_bird`).

3 Bird density profiles

Bird density profiles were determined from all available data from the 62 radar systems described in this report, for the entire test period.

The entire data set was first processed using a cluttermap using the zero percentile (no extra clutter filtering). An example of the reflectivities used for the processing is shown in Figure 3.1. In this figure a composite ppi plot was created using the reflectivity fields of the lowest scan of each radar systems. Note that the lowest scans do not necessarily refer to the same heights, as the radar heights and the scan angles may vary between radar systems. The ppi's are shown for two moments in time: 15 Sep. 2011 at 17:00 UTC and 15 Sep. 2011 at 18:30 UTC.

In Figure 3.2 the same composite ppi's are shown as before, but now the data is 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 atributed to rain and/or clutter and most of the signal is removed. However, from a comparison of the filtered ppi's at 17:00 UTC and 18:30 UTC (cf. top and bottom panel in Figure 3.2) an intensification of reflectivities can be observed in central Europe, i.e. Poland, Slovakia and Slovenia.

In Figures 3.3-3.5 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 cluttermaps 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 cluttermap definition of 25% and settings for small passerines (see *Dokter et al.* (2011), eq. 2.15). These can be found in directory `bird`.

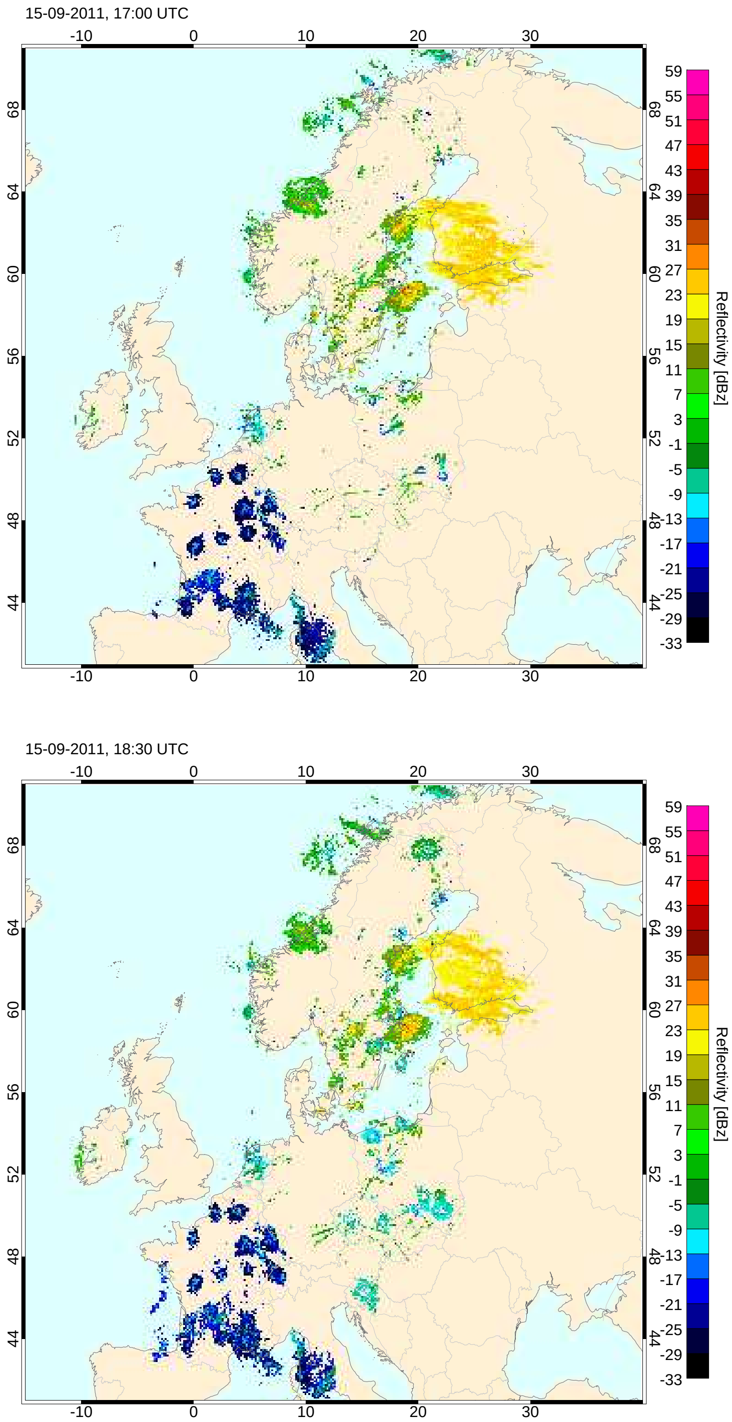
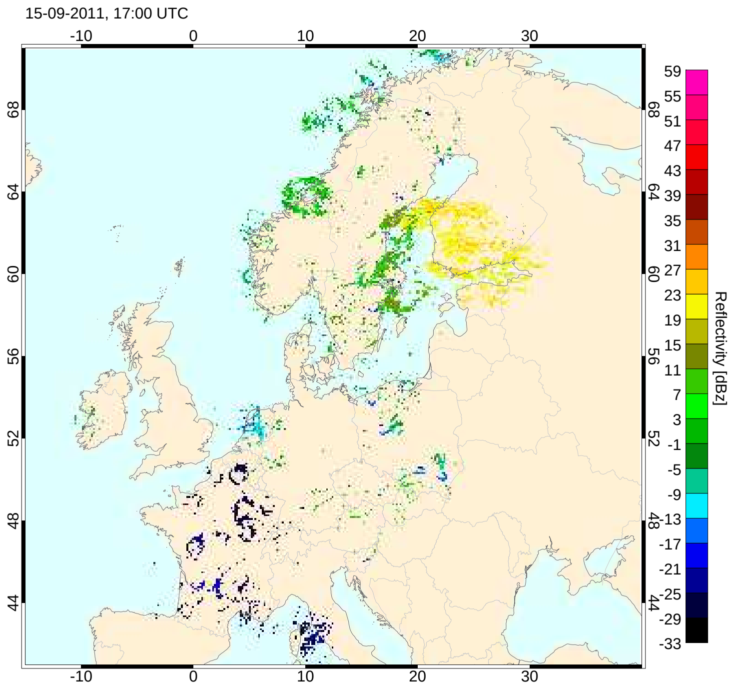


Figure 3.1. PPI composite of reflectivities of the lowest scans of all stations on 15 Sep. 2011 at 17:00 UTC (top panel) and 18:30 UTC (bottom panel).



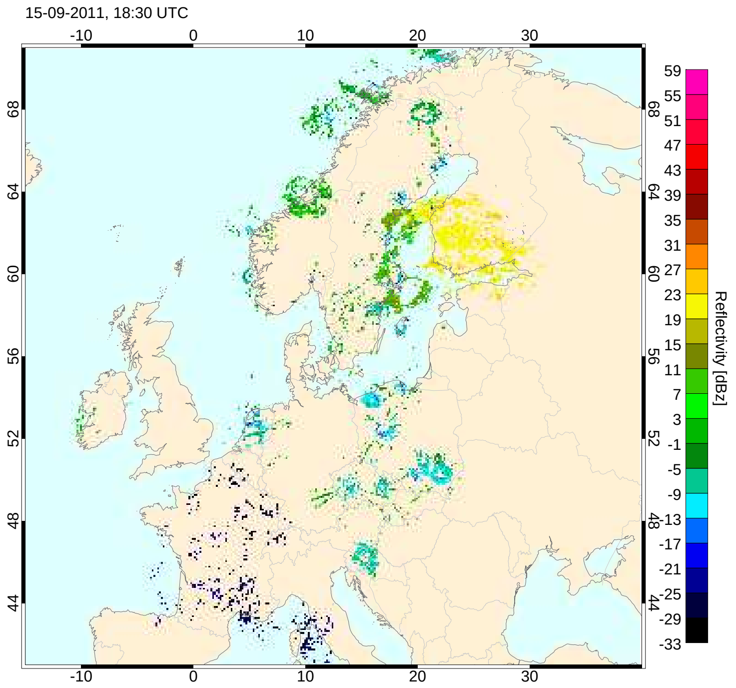


Figure 3.2. PPI composite of reflectivities of the lowest scans of all stations on 15 Sep. 2011 at 17:00 UTC (top panel) and 18:30 UTC (bottom panel), filtered for rain cells.

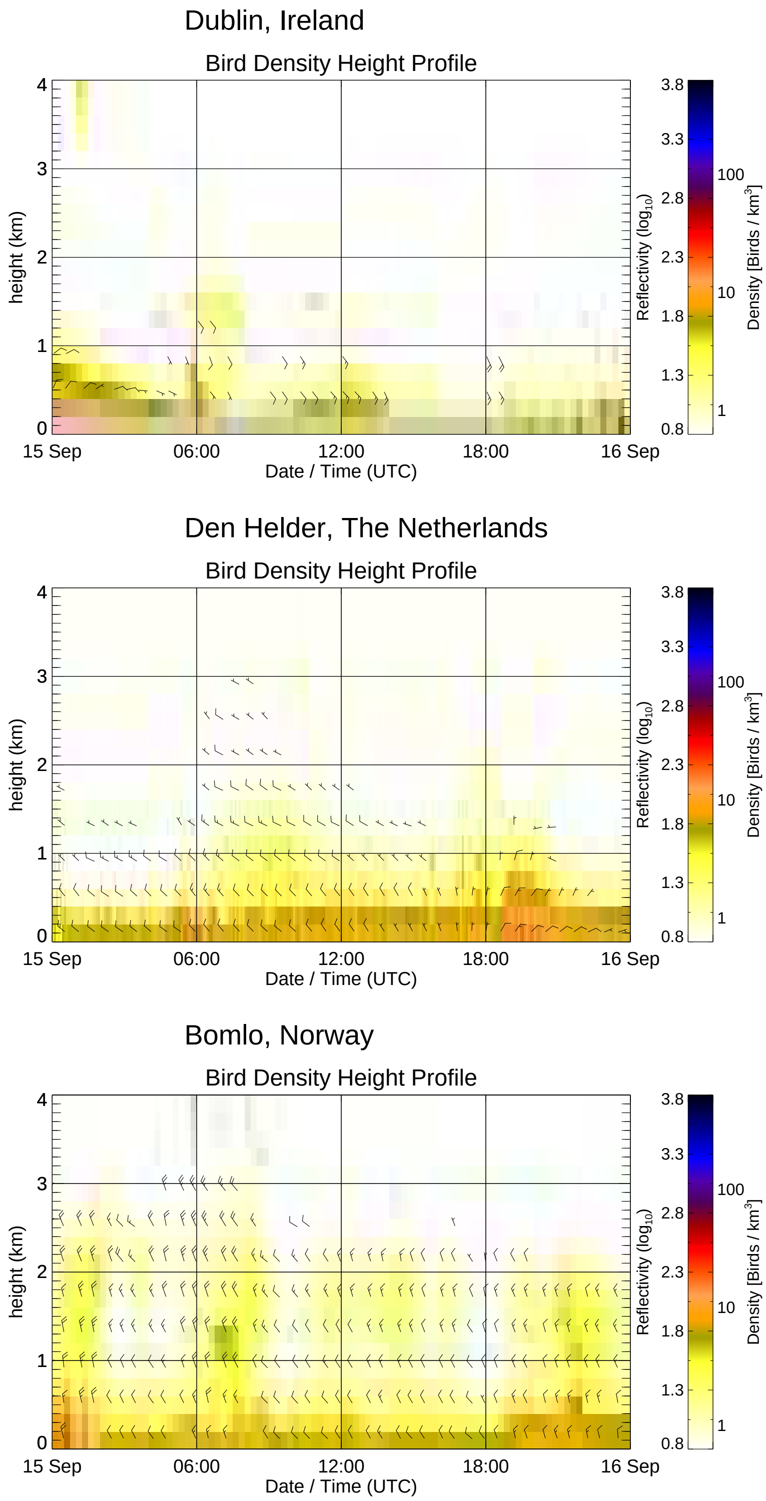


Figure 3.3. Bird density profiles from 0 - 4 km altitude during 15 Sep. 2011 as a function of time, averaged from 5 - 25 km distance of the indicated radar system.

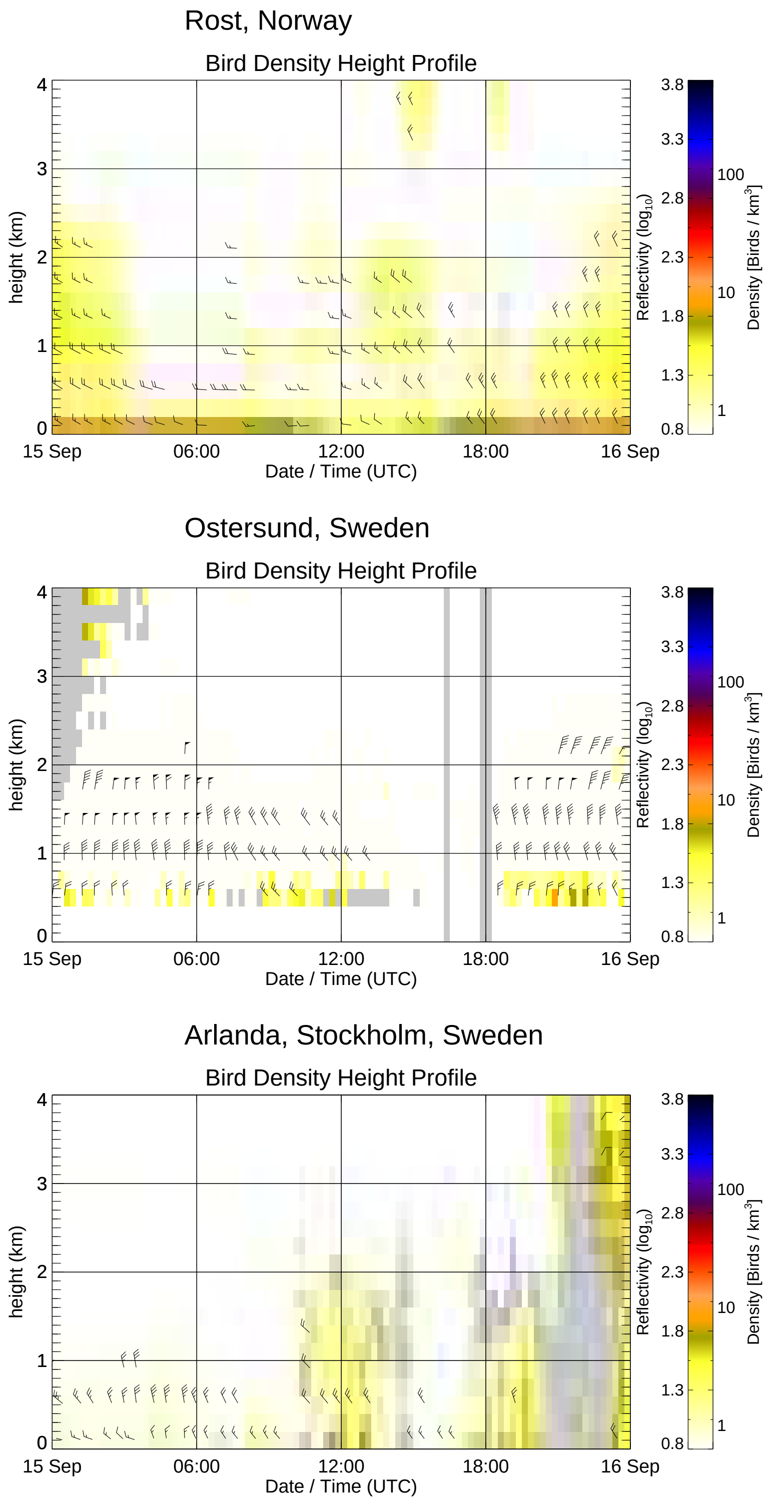


Figure 3.4. Bird density profiles from 0 - 4 km altitude during 15 Sep. 2011 as a function of time, averaged from 5 - 25 km distance of the indicated radar system.

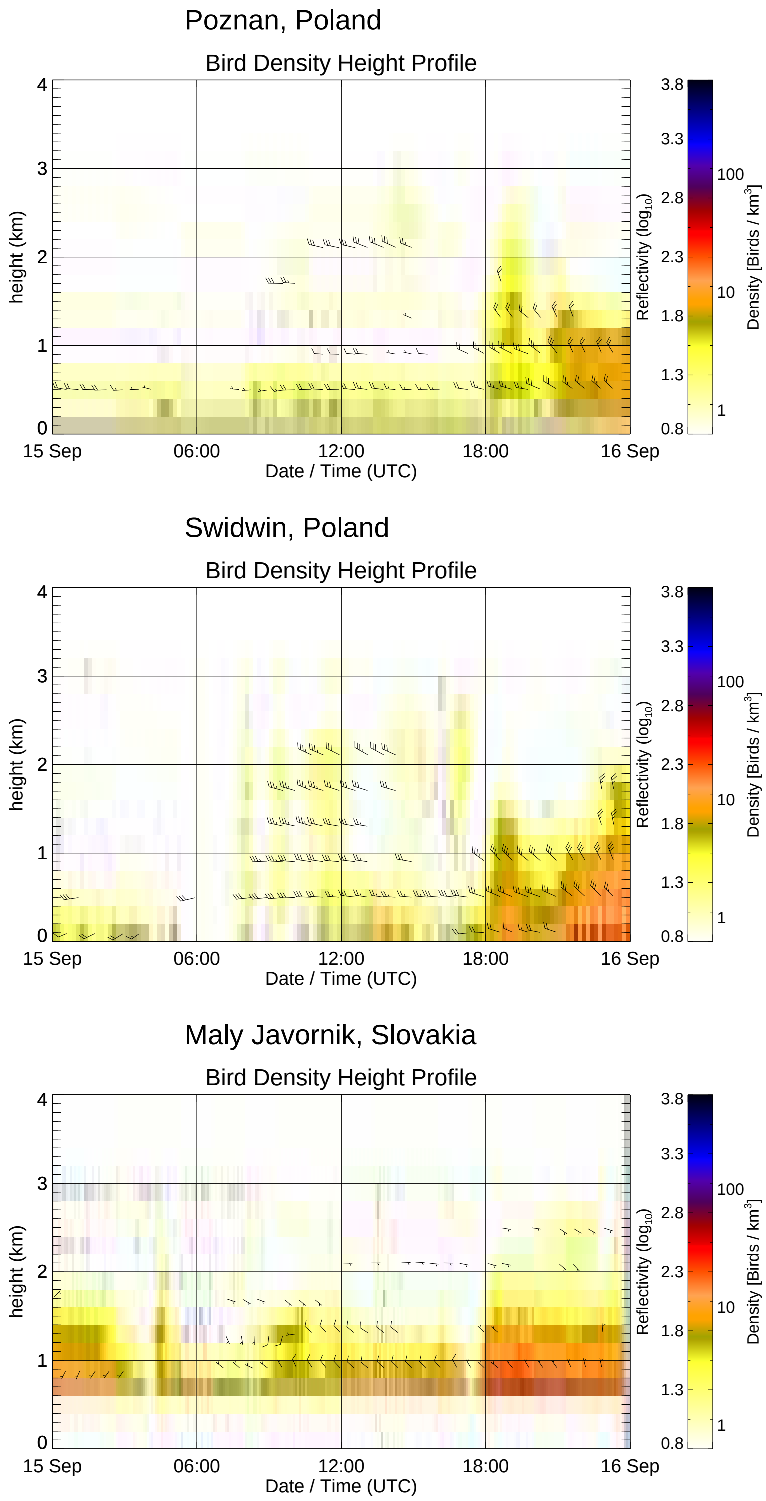


Figure 3.5. Bird density profiles from 0 - 4 km altitude during 15 Sep. 2011 as a function of time, averaged from 5 - 25 km distance of the indicated radar system.

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 IO-procedures, which provide the interface to the actual data. These IO-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 IO-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 IO-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 IO-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 recognisable format. The available IO-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 IO-procedures all routines start with the convention `vol2bird_cc_dataformat_command`. E.g. the command for reading a scan

CC	Country	IO-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 IO-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 IO-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 capitales, an _, and a three element string abbreviations of the actual name. `IO_FILE` points to the IO-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 clutternmaps 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 sytems, 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-polarised total uncorrected reflectivity factor (TH), if available, the logged horizontally-polarised 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 recognisable 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.