Radar and SAFNWC data APIs in SHMI

Ladislav Méri
Ľuboslav Okon

SHMI, Bratislava, Slovakia
6 – 7 February, 2012

(API = *Application Programming Interface*)

Why radar volumes and SAFNWC?

Goals:

- Improve the quantitative precipitation estimation
- Improve the precipitation nowcasting

Our solution:

- Transition from 2D to 3D
- Quality index based precipitation analysis

Actual input data:

- text files with a 2D composite of some standard radar product (CAPPI 2km, BASE, …)

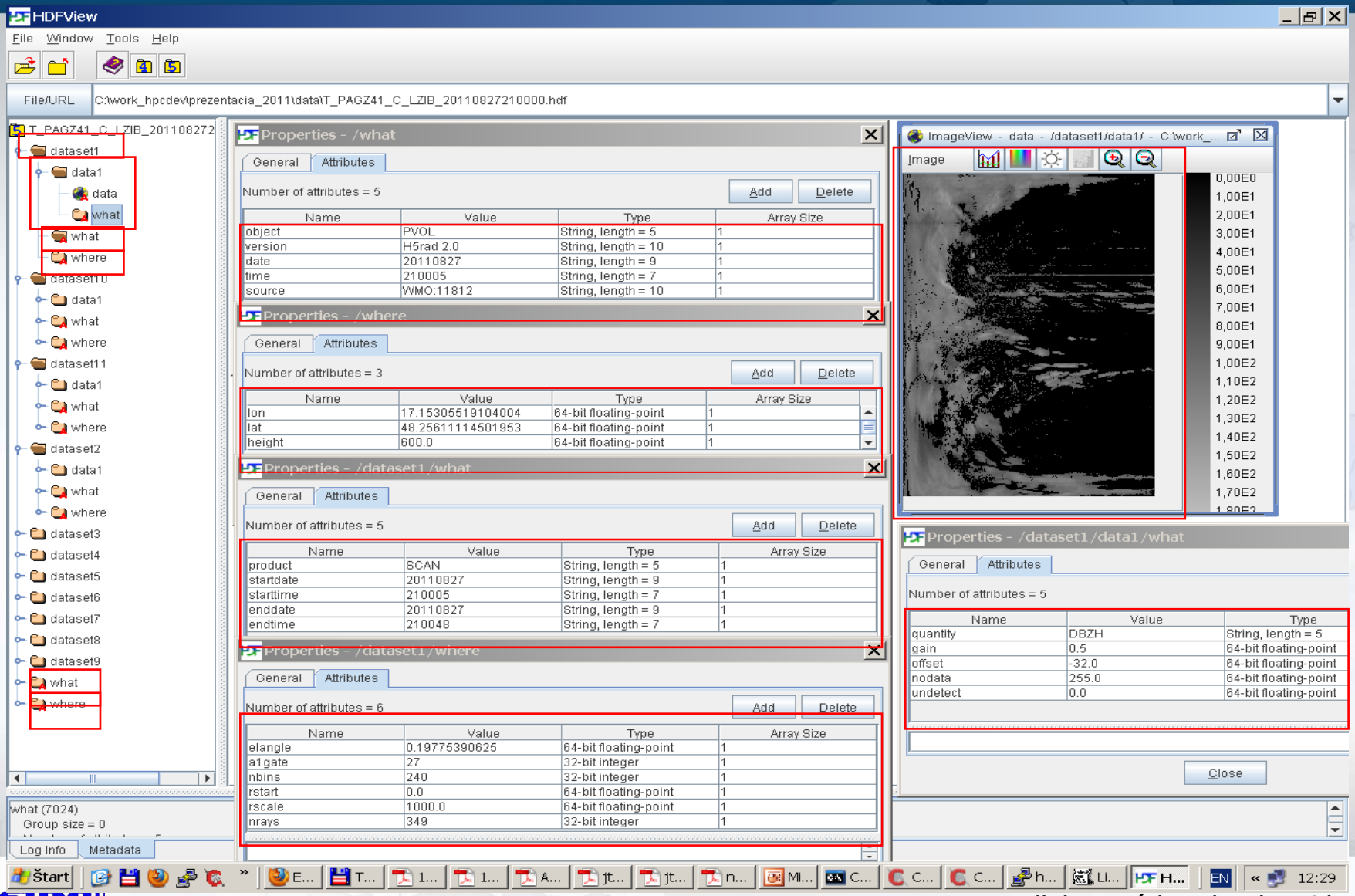
New input data:

- volume data from several radars in ODIM_H5 format (OPERA Data Information Model for HDF5) and SAFNWC products (mainly in HDF5)

Advantages:

- flexibility (selection of required radars, interpolation and compositing)
- product generation (CAPPI, CMAX, BASE, ECHOTOP, VIL, …)
- quality control (quality indexes, data check and filtering)

ODIM H5


HDFView
File Window Tools Help
File/URL
C:\work_hpcdev\prezentacia_2011\data\T_PAGZ41_C_LZIB_20110827210000.hdf
Properties - /what
Properties - /where
Properties - /dataset1/what
Properties - /dataset1/where
Properties - /dataset1/data1/what
ImageView - data - /dataset1/data1/ - C:\work_...
Image
Close
Log Info
Metadata

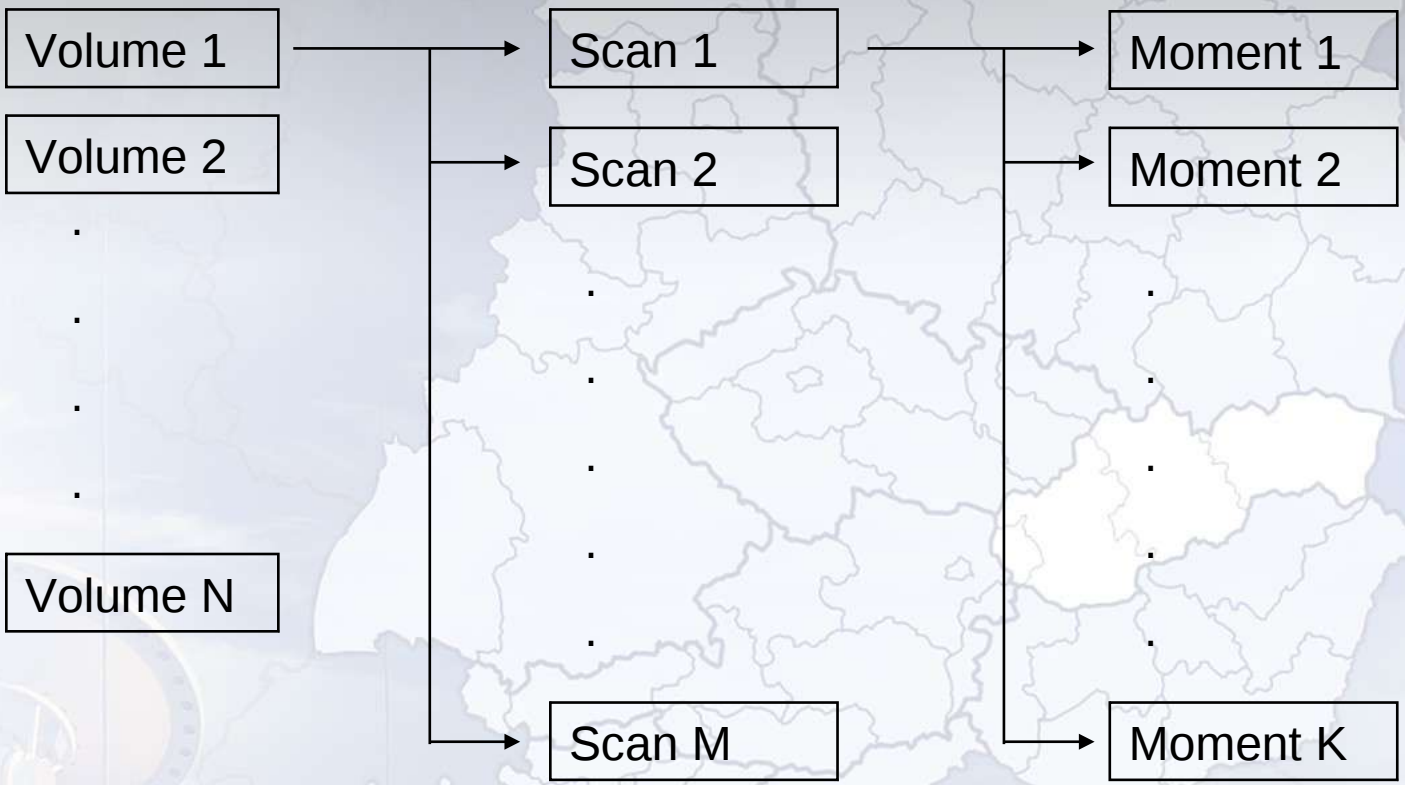
Module my_hdf5

```
/*this module uses the hdf5 api functions (H5Fopen, H5Gopen, H5Dread, …)*/
int read_double_attribute( hid_t Where, char *Name, double *Value );
int read_float_attribute( hid_t Where, char *Name, float *Value );
int read_int_attribute( hid_t Where, char *Name, int *Value );
int read_string_attribute( hid_t Where, char *Name, char *Value );
char *read_2D_char_data( hid_t Where, char *Name, int *DataSize );
unsigned short *read_2D_ushort_data( hid_t Where, char *Name, int *DataSize );
```

Example:

```
/*open the file*/
Hid_t FileID, GrID;
FileID = H5Fopen("/opera/20110607/cz/T_PAGZ50_C_OKPR_20110607170000.hdf", H5F_ACC_RDONLY, H5P_DEFAULT );
   if ( FileID < 0 ) {
     return -1;
   }
/*open the “/where” group*/
   GrID = H5Gopen( FileID, “/where”, H5P_DEFAULT );
   if ( GrID < 0 ) {
     return -1;
   }
/*read some parameters*/
Double Lat, Lon; int err;
     err = read_double_attribute( GrID, "lat", &Lat );
     err = read_double_attribute( GrID, "lon", &Lon );
     printf(“Radar Lat: %f, Lon %f\n”, Lat, Lon);
```

Radar volume data structure



Module radar moment

```
typedef struct radar_moment {
    char    Quantity[32];       /* moment name (TH, DBZH, VRAD, ...) */
    int     RayNumber;          /* number of rays */
    int     BinNumber;          /* number of bins */
    double  Gain;               /* bin_value = Gain*BScope[i][j] + Offset */
    double  Offset;
    unsigned char  NoData;      /* value for never radiated area     */
    unsigned char  Undetect;    /* radiated, but nothing detected */
    unsigned char *BScope;      /* array of the meassured data in b-scope mode */
 } RADAR_MOMENT;

int radar_moment_init( RADAR_MOMENT *Mom, char *Quantity, int RayNumber, int BinNumber, double Gain, double Offset,
    unsigned char NoData, unsigned char Undetect );
int radar_moment_deinit( RADAR_MOMENT *Mom );
int radar_moment_set_bin_char( RADAR_MOMENT *Mom, int RayI, int BinI, unsigned char BinVal );
int radar_moment_set_bin_doub( RADAR_MOMENT *Mom, int RayI, int BinI, double BinVal );
int radar_moment_get_bin_doub( RADAR_MOMENT *Mom, int RayI, int BinI, double *BinVal );
int radar_moment_copy( RADAR_MOMENT *MomIn, RADAR_MOMENT *MomOut );
...
```

Module radar scan

```
typedef struct radar_scan {
   char   StartDate[9];     /* starting YYYYMMDD */
   char   StartTime[7];     /* starting hhmmss */
   char   EndDate[9];       /* ending YYYYMMDD */
   char   EndTime[7];       /* ending hhmmss */
   double ElAngle;          /* antenna elevation angle (degrees) above the horizon */
   double RStart;           /* the range (km) of the start of the first range bin */
   double RScale;           /* the distance in meters between two successive range bins */
   int    NRays;            /* number of azimuth gates (rays) in the scan */
   int    NBins;            /* number of range bins in each ray */
   int    A1Gate;           /* index of the first azimuth gate radiated in the scan (1-NRays)*/
   int    MomentNumber;     /* number of moments */
   RADAR_MOMENT *Moments;   /* list of moments in the scan */
} RADAR_SCAN;

int radar_scan_init( RADAR_SCAN *Scan, char *StartDate, char *StartTime, char *EndDate, char *EndTime,
                     double ElAngle, double RStart, double RScale, int NRays, int NBins, int A1Gate );
int radar_scan_add_moment( RADAR_SCAN *Scan, RADAR_MOMENT *Mom );
int radar_scan_rayindex2azi( RADAR_SCAN *Scan, int index, double *Azi );
int radar_scan_azi2rayindexpair( RADAR_SCAN *Scan, double Azi, int *IndexPair );
int radar_scan_get_bin_value_by_moment_index( RADAR_SCAN *Scan, int MomentI, int RayI, int BinI, double *BinVal );
int radar_scan_get_bin_value_by_moment_quantity( RADAR_SCAN *Scan, char *Qua, int RayI, int BinI, double *BinVal );
```

Module radar volume

```
typedef struct radar_volume {
    char   Date[9];          /* Nominal Year, Month, and Day of the data (YYYYMMDD) */
    char   Time[7];          /* Nominal Hour, Minute, and Second, in UTC of the data (hhmmss) */
    char   Source[128];      /* Radar site indetifier */
    double Lat;              /* Latitude position of the radar antenna (degrees) */
    double Lon;              /* Longitude position of the radar antenna (degrees) */
    double Height;           /* Height of the centre of the antenna in meters above sea level */
    double Beamwidth;        /* The radar's half-power beamwidth (degrees) */
    int    ScanNumber;       /* Number of elevation scans */
    RADAR_SCAN *Scans;       /* List of scans in the volume */
} RADAR_VOLUME;

int radar_volume_add_scan( RADAR_VOLUME *Vol, RADAR_SCAN *Scan );
int radar_volume_add_moment( RADAR_VOLUME *Vol, RADAR_MOMENT *Mom, int ScanIndex );
int radar_volume_azi2rayindexpair( RADAR_VOLUME *Vol, int ScanIndex, double Azi, int *IndexPair );
int radar_volume_binindex2range( RADAR_VOLUME *Vol, int ScanIndex, int BinIndex, double *Range );
int radar_volume_scanindex2elev( RADAR_VOLUME *Vol, int i, double *Val );
int radar_volume_elev2scanindexes( RADAR_VOLUME *Vol, double ElAngle, int *IndexNum, int *IndexList );
int radar_volume_get_bin_value_by_moment_index( RADAR_VOLUME *Vol, int ScanI, int MomI, int RayI, int BinI,
                                                double *BinVal );
int radar_volume_get_bin_value_by_moment_quantity( RADAR_VOLUME *Vol, int ScanI, char *Qua, int RayI, int
BinI, double *BinVal );
```

Reading radar volume from an ODIM_H5 file

- module_myhdf5 + module_radar_volume = module_radar_volume_io:

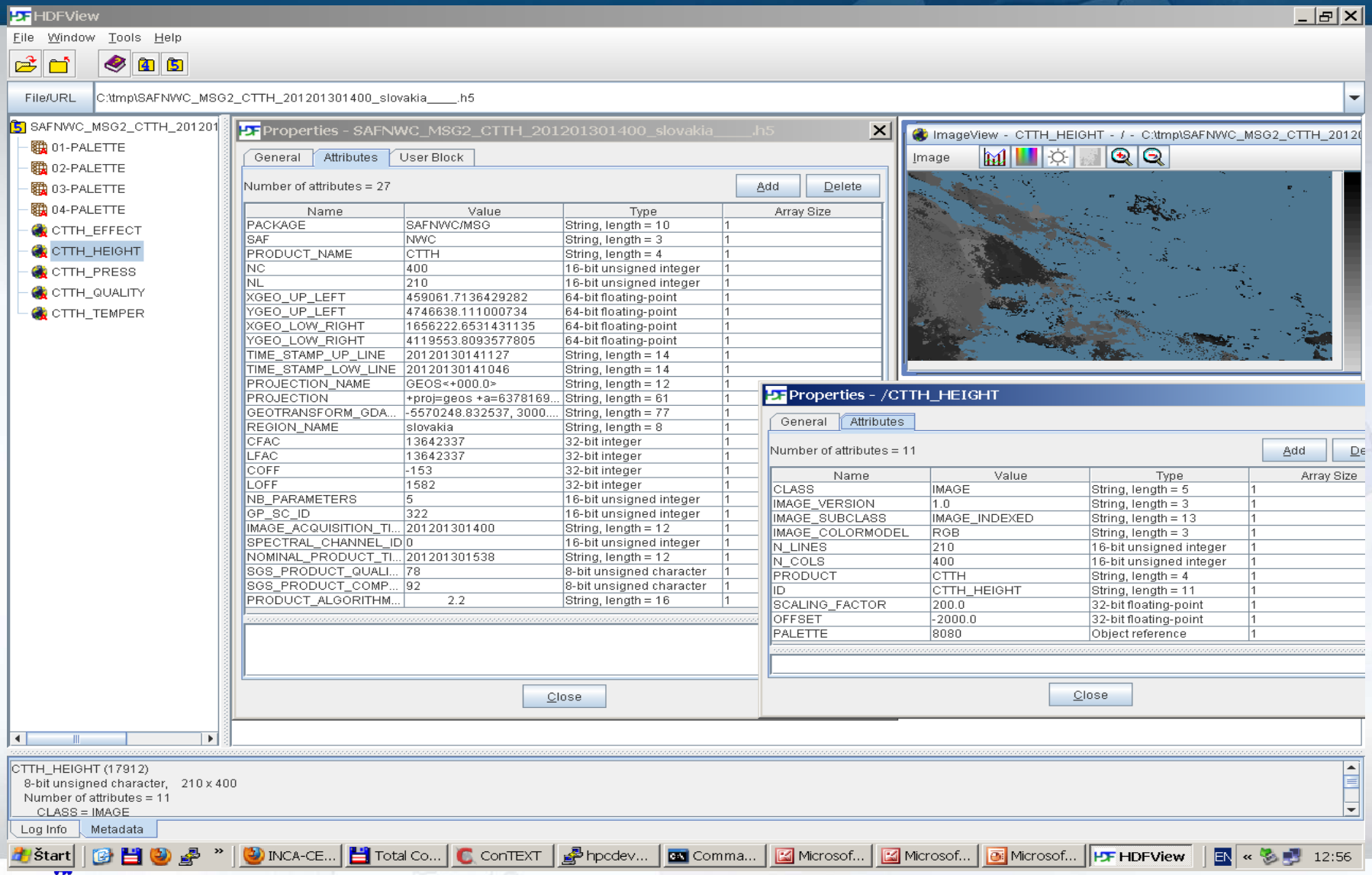
```
int radar_moment_read_by_index_from_odimh5( RADAR_MOMENT *Mom, char *FileName, int ScanIndex,
                                            int MomentIndex );
int radar_scan_read_by_index_from_odimh5( RADAR_SCAN *Scan, char *FileName, int ScanIndex );
int radar_volume_read_from_odimh5( RADAR_VOLUME *Vol, char *FileName );
```

- example:

```
RADAR_VOLUME Vol; int err;
err = radar_volume_read_from_odimh5( &Vol, "/opera/20110607/cz/T_PAGZ50_C_OKPR_20110607170000.hdf" );
if ( err != 0 ) {
  return -1;
}
printf("radar latitude: %f, radar longitude: %f \n", Vol1.Lon, Vol1.Lat);

int scani = 1, momi = 1, rayi = 183, bini = 135;
double d;
err = radar_volume_get_bin_value_by_moment_index( &Vol, scani, momi, rayi, bini, &d );
printf("The bin %d in scan %d, moment %d, ray %d has a value of %f\n", bini, scani, momi, rayi, d);
```

SAFNWC hdf5


HDFView
File Window Tools Help
File/URL
C:\tmp\SAFNWC_MSG2_CTTH_201201301400_slovakia____.h5
SAFNWC_MSG2_CTTH_201201
01-PALETTE
02-PALETTE
03-PALETTE
04-PALETTE
CTTH_EFFECT
CTTH_HEIGHT
CTTH_PRESS
CTTH_QUALITY
CTTH_TEMPER
Properties - SAFNWC_MSG2_CTTH_201201301400_slovakia____.h5
General Attributes User Block
Number of attributes = 27
Add Delete
Name Value Type Array Size
PACKAGE SAFNWC/MSG String, length = 10 1
SAF NWC String, length = 3 1
PRODUCT_NAME CTTH String, length = 4 1
NC 400 16-bit unsigned integer 1
NL 210 16-bit unsigned integer 1
XGEO_UP_LEFT 459061.7136429282 64-bit floating-point 1
YGEO_UP_LEFT 4746638.111000734 64-bit floating-point 1
XGEO_LOW_RIGHT 1656222.6531431135 64-bit floating-point 1
YGEO_LOW_RIGHT 4119553.8093577805 64-bit floating-point 1
TIME_STAMP_UP_LINE 20120130141127 String, length = 14 1
TIME_STAMP_LOW_LINE 20120130141046 String, length = 14 1
PROJECTION_NAME GEOS<+000.0> String, length = 12 1
PROJECTION +proj=geos +a=6378169... String, length = 61 1
GEOTRANSFORM_GDA... -5570248.832537, 3000.... String, length = 77 1
REGION_NAME slovakia String, length = 8 1
CFAC 13642337 32-bit integer 1
LFAC 13642337 32-bit integer 1
COFF -153 32-bit integer 1
LOFF 1582 32-bit integer 1
NB_PARAMETERS 5 16-bit unsigned integer 1
GP_SC_ID 322 16-bit unsigned integer 1
IMAGE_ACQUISITION_TI... 201201301400 String, length = 12 1
SPECTRAL_CHANNEL_ID 0 16-bit unsigned integer 1
NOMINAL_PRODUCT_TI... 201201301538 String, length = 12 1
SGS_PRODUCT_QUALI... 78 8-bit unsigned character 1
SGS_PRODUCT_COMP... 92 8-bit unsigned character 1
PRODUCT_ALGORITHM... 2.2 String, length = 16 1
Close
ImageView - CTTH_HEIGHT - / - C:\tmp\SAFNWC_MSG2_CTTH_2012
Image
Properties - /CTTH_HEIGHT
General Attributes
Number of attributes = 11
Add
Name Value Type Array Size
CLASS IMAGE String, length = 5 1
IMAGE_VERSION 1.0 String, length = 3 1
IMAGE_SUBCLASS IMAGE_INDEXED String, length = 13 1
IMAGE_COLORMODEL RGB String, length = 3 1
N_LINES 210 16-bit unsigned integer 1
N_COLS 400 16-bit unsigned integer 1
PRODUCT CTTH String, length = 4 1
ID CTTH_HEIGHT String, length = 11 1
SCALING_FACTOR 200.0 32-bit floating-point 1
OFFSET -2000.0 32-bit floating-point 1
PALETTE 8080 Object reference 1
Close
CTTH_HEIGHT (17912)
8-bit unsigned character, 210 x 400
Number of attributes = 11
CLASS = IMAGE
Log Info Metadata
Start
INCA-CE... Total Co... ConTEXT hpcdev... Comma... Microsof... Microsof... Microsof... HDFView
EN
12:56

Module map array

```
typedef struct map_array {
    int     nx;             /*number of points in x direction*/
    int     ny;             /*number of points in y direction*/
    int     nz;             /*number of points in z direction*/
    int     IsInit;
    double *Data;           /*poi. to the data array*/
} MAP_ARRAY;

int map_array_init( MAP_ARRAY *MA, int nx, int ny, int nz, double IniVal );
int map_array_deinit( MAP_ARRAY *MA );
int map_array_get_value( MAP_ARRAY *MA, int x, int y, int z, double *Val );
int map_array_set_value( MAP_ARRAY *MA, int x, int y, int z, double Val );
```

Module geos map

```
typedef struct geos_map {
    int    Nx;              /*number of columns (pixels in x direction)*/
    int    Ny;              /*number of lines (pixels in y direction)*/
    double SubLon;          /*sub-satellite longitude in degrees*/
    double coff;            /*column offset*/
    double cfac;            /*column scale factor (c = coff + x*cfac*(2^-16)), c(1-Nx)*/
    double loff;            /*line offset*/
    double lfac;            /*line scale factor (l = lf + y*lfac*(2^-16)), l(1-Ny)*/
    double cfac2;           /*cfac*(2^-16)*/
    double lfac2;           /*lfac*(2^-16)*/
    MAP_ARRAY Map;          /*map array*/
} GEOS_MAP;

int geos_map_init( GEOS_MAP *GMap, int Nx, int Ny, double SubLon, double coff, double cfac, double loff,
                   double lfac, double IniVal );
int geos_map_deinit( GEOS_MAP *GMap );
int geos_map_lonlat2xy( GEOS_MAP *GMap, double lon, double lat, int *x, int *y );
int geos_map_xy2lonlat( GEOS_MAP *GMap, int x, int y, double *lon, double *lat );
int geos_map_set_value_by_index( GEOS_MAP *GMap, int x, int y, double Val );
int geos_map_get_value_by_index( GEOS_MAP *GMap, int x, int y, double *Val );
int geos_map_set_value_by_coord( GEOS_MAP *GMap, double lon, double lat, double Val );
int geos_map_get_value_by_coord( GEOS_MAP *GMap, double lon, double lat, double *Val );
```

Reading SAFNWC products

- module_my_hdf5 + module_geos_map = module_geos_map_io:

```
int geos_map_read_from_safnwch5( GEOS_MAP *GMap, char *H5FileName, char *ProductName, int ByteSize );
```

- Example:

```
GEOS_MAP CTMap; int err;
err = geos_map_read_from_safnwch5( &CTMap, "SAFNWC_MSG2_CT___201201191600_slovakia____.h5", "CT", 1 );
printf("CT read: %d\n", err);

double d;
int x = 100, y = 100;
err = geos_map_get_value_by_index(&CTMap, x, y, &d );
printf(“The Cloud Type value in column %d and line %d is %f\n”, x, y, d);

err = geos_map_get_value_by_coord(&CTMap, 17.151944, 48.255, &d );
printf(“The Cloud Type value in point (Lon = 17.151944, Lat = 48.255) is %f\n”, d);
```

Radar bin coordinates

```
typedef struct radar_bin_coord {
    int ScanNumber;
    int RayNumber;
    int BinNumber;
    float *LonTable;
    float *LatTable;
    float *HeiTable;
    RADAR_VOLUME *Rad;
} RADAR_BIN_COORD;

int radar_bin_coord_init( RADAR_BIN_COORD *RbcTable, RADAR_VOLUME *Vol );

int radar_bin_coord_compute_from_eq_earth( RADAR_BIN_COORD *RbcTable, RADAR_VOLUME *Vol );
int radar_bin_coord_compute_from_beam_prop_model( RADAR_BIN_COORD *RbcTable, RADAR_VOLUME *Vol,
                                                  LAMBERT_MAP *n_grad );

int radar_bin_coord_get_closest_bin_indexes( RADAR_BIN_COORD *RbcTable,  double Plon, double Plat,
                                             double Phei, INT_LIST *ScanIndexList, INT_LIST *RayIndexList, INT_LIST *BinIndexList );
```

Radar Quality Check

- Quality Index (value from 0.0 to 1.0, 1=good quality, 0=bad quality) computing for every bin:

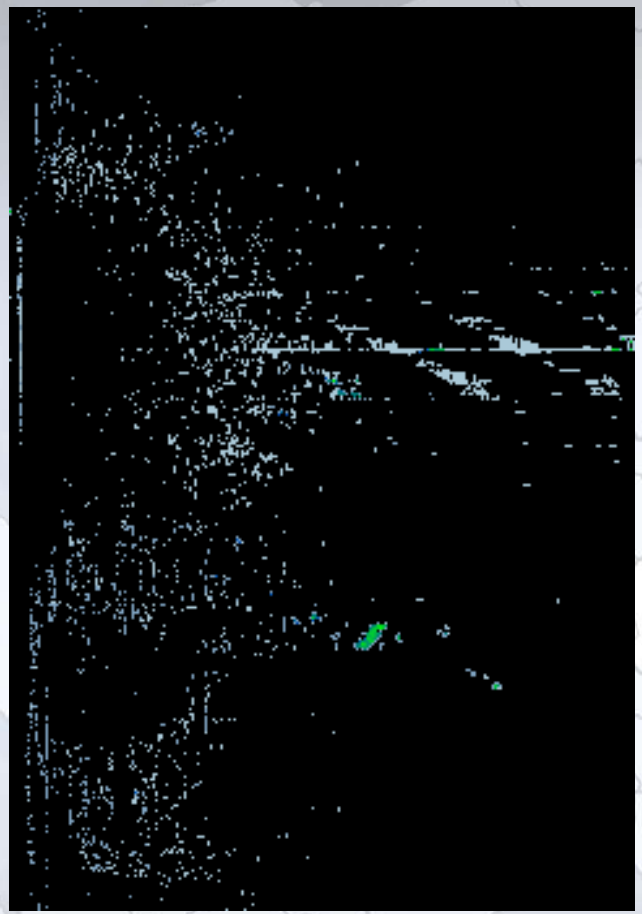
```
int single_radar_qi_set_laplace_for_volume( RADAR_VOLUME *Vol, char *MomName, char
    *QIName, double MinTH, double MaxTH );

int single_radar_qi_set_rlan_for_volume( RADAR_VOLUME *Vol, char *MomName, char
    *QIName, double StartTH, double AmpTH, int MaxWidth );

int single_radar_qi_set_atten_for_volume( RADAR_VOLUME *Vol, char *MomName, char
    *QIName, double a, double b, double MaxF, char *QIFilt1Name, double QIFilt1Thr, char
    *QIFilt2Name, double QIFilt2Thr );

int single_radar_qi_set_safnwc_for_volume( RADAR_VOLUME *Vol, char *MomName,
    RADAR_BIN_COORD *RadBinCo, GEOS_MAP *CTMap, GEOS_MAP *CTHMap,
    GEOS_MAP *PCMap, char *QIName );
```

Radar Quality Check - Example



Radar Maly Javornik

Module inca_proj_map

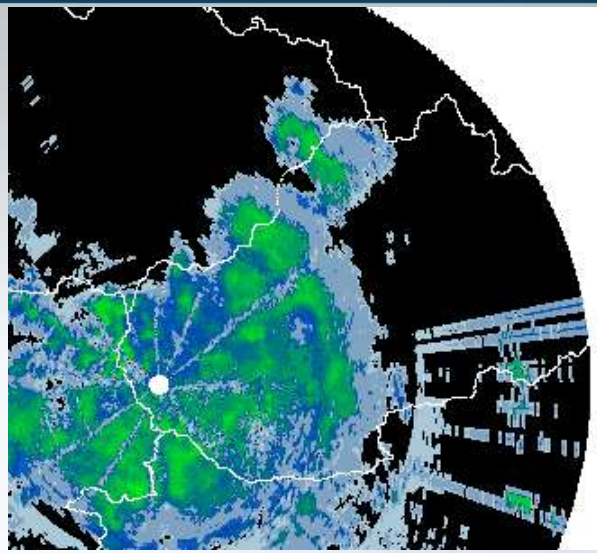
```
typedef struct inca_proj_map {
    double x0;          /*x for SW corner from subroutine KEKO (see file inca_proj.f)*/
    double y0;          /*y for SW corner from subroutine KEKO (see file inca_proj.f)*/
    double z0;          /*height for the lowest level in [m] ASL*/
    double dx;          /*step in x direction          (x = x0 + (i-1)*dx  ->  for i = 1,....,nx)*/
    double dy;          /*step in y direction          (y = y0 + (j-1)*dy  ->  for j = 1,....,ny)*/
    double dz;          /*step in vertical dir. in [m]*/
    int   nx;          /*point number in x direction*/
    int   ny;          /*point number in y direction*/
    int   nz;          /*nuber of verical levels*/
    MAP_ARRAY Map;          /*map array*/
  } INCA_PROJ_MAP;
```

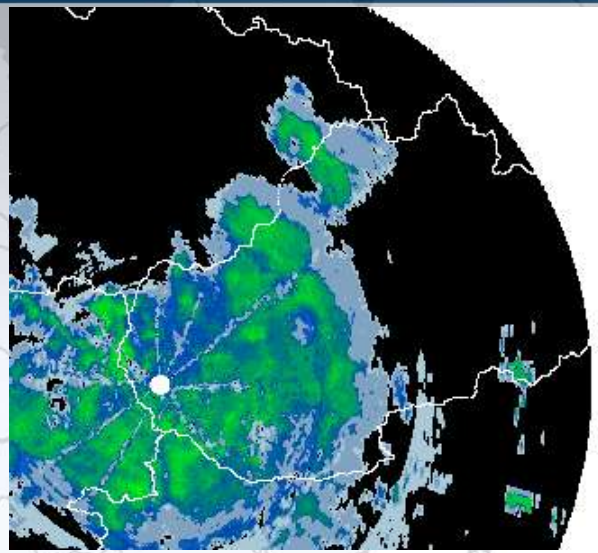
```
int inca_proj_map_init( INCA_PROJ_MAP *IPMap, double x0, double y0, double z0, double dx, double dy, double dz, int
    nx, int ny, int nz, double IniVal );
int inca_proj_map_LonLatHeight2xyz( INCA_PROJ_MAP *IPMap, double Lon, double Lat, double Hei, int *x, int *y, int *z
    );
int inca_proj_map_xyz2LonLatHeight( INCA_PROJ_MAP *IPMap, int x, int y, int z, double *Lon, double *Lat, double *Hei
    );
int inca_proj_map_set_value_by_index( INCA_PROJ_MAP *IPMap, int x, int y, int z, double Val );
int inca_proj_map_get_value_by_coord( INCA_PROJ_MAP *IPMap, double Lon, double Lat, double Hei, double *Val );
```

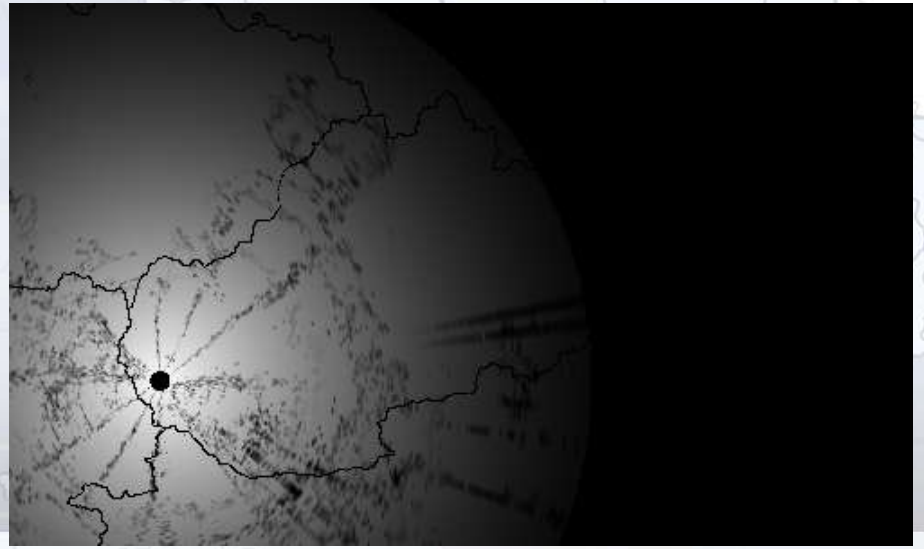
Example with radar volume

```
/*read the radar volume from a hdf5 file*/
RADAR_VOLUME Vol; int err;
err = radar_volume_read_from_odimh5( &Vol, "/opera/20110607/cz/T_PAGZ50_C_OKPR_20110607170000.hdf" );
/*set some quality indexes*/
err = single_radar_qi_set_rlan_for_volume( &Vol1, "DBZH", "QI_RLAN", 4.0, 8.0, 8.0 );
err = single_radar_qi_set_laplace_for_volume( &Vol1, "DBZH", "QI_LAPL", 50.0, 150.0 );
err = single_radar_qi_set_atten_for_volume( &Vol1, "DBZH", "QI_ATTEN", 200.0, 1.6, 3.0, "QI_RLAN", 0.5, "QI_LAPL", 0.5  );
/*init. and set the radar bin coordinate table*/
RADAR_BIN_COORD RadBinCo;
err = radar_bin_coord_init( &RadBinCo, &Vol1 );
err = radar_bin_coord_compute_from_eq_earth( &RadBinCo, &Vol1 );
/*init. the output INCA maps*/
INCA_PROJ_MAP DbzhMap, QiMap;
err = inca_proj_map_init( &DbzhMap, 600.0, 400.0, 1000.0, 1.0, 1.0, 1000.0, 501, 301, 9, 95.5 );
err = inca_proj_map_init( &QiMap, 600.0, 400.0, 1000.0, 1.0, 1.0, 1000.0, 501, 301, 9, 0.0 );
/*put the radar volume in to the maps*/
err = opera2inca_vol2map3D( &Vol1, &RadBinCo, 300000.0, "DBZH", &DbzhMap, &QiMap );
/*read some data from the map*/
int x = 100, y = 120, z = 2;
err = inca_proj_map_get_value_by_index( &DbzhMap, x, y, z, &d );
printf("DBZH value in point %d %d %d is %f.\n", x, y, z, d);
```

3D radar map example







Future plans

- Implement other quality indexes for radar measurement (beam blockage, …)
- Compute quality indexes of rainfall estimation from the radar data (height of the level used for estimate the rainfall, accuracy of Marshall-Palmer parameters, possible bright band, uncertainties in the phase of the rainfall (snow, liquid, hail, mixed))
- Compute quality indexes also for rain gauges data
- Combine all the measurements (rain gauge, radar, SAFNWC, HSAF, …) to get the best estimation of the rainfall rate

Thank You!

ladislav.meri@shmu.sk, luboslav.okon@shmu.sk