INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2015, VOL. 61, NO. 4, PP. 345–350
Manuscript received October 15, 2015; revised November, 2015. DOI: 10.1515/eletel-2015-0045

Application for Displaying Synthetic Aperture Radar Imagery in Real-time Implementation and Results

Krzysztof Borowiec

***Abstract*—The paper presents implementation and results of the application for displaying SAR (Synthetic Aperture Radar) imagery operating in real-time. The application performs SAR imagery formation and displays results in real-time after receiving of preprocessed data via an SAR processing application. The application was used in SARape (Synthetic Aperture Radar for all weather penetrating UAV application) project founded by the European Defence Agency. The real-time operation is achieved thanks to implementation based on multithreading.**

***Keywords*—synthetic aperture radar, real-time imaging**

I. Introduction

THIS paper presents a real-time SAR (Synthetic Aperture Radar) displaying application developed at the Warsaw University of Technology. Originally the application development was started in the framework of the SARape (Synthetic Aperture Radar for all weather penetrating UAV application) [2] [3] project, sponsored by the European Defence Agency. After the SARape project was concluded the development of the application continued, which has resulted in numerous new features and functionalities.

The original operating scenario was defined in the SARape project. The aim of the project was to develop a high resolution SAR demonstrator for UAVs (Unmanned Aerial Vehicles). The radar operates in FMCW (Frequency- Modulated Continuous-Wave) mode, the analog front-end module sends and receives the radar signal, and the signal from two output channels are sampled, down-converted, filtered and decimated in the preprocessing module. After this, the signal is sent in the form of UDP (User Datagram Protocol) packets to the telemetry module with a data stream of up to 80 Mbit/s. On the ground, the received data are then sent using UDP packets to the real-time SAR processor, which uses sophisticated signal processing algorithms for image formation and motion compensation. The core part of the processing is realized on a graphics card using CUDA (Compute Unified Device Architecture) technology [4]. The resulting SAR image is sent to the displaying application in the form of UDP packets.

According to the assumptions made in the SARape project, the SAR processor should receive the data from two channels.Therefore, the application should visualize data from both of them. The data received from the processing application are in the form of a two dimensional image, the dimensions corresponding to the range and cross-range dimensions. In one of the displaying modes, the so called simple mode, the image is displayed directly in range/cross-range coordinates. In this mode, however, the orientation and geographic position of the platform are not taken into account. In the second displaying mode, the so called world coordinate mode, the data are transformed so that they can be overlaid on a map.

The main characteristics of the application can be summarized as follows:

- Provides clear and reliable graphical user interface.
- Provides two modes of SAR image display simple mode (range/cross-range display) and world coordinate mode (geographically oriented display).
- Displays vector map with a legend (rivers, lakes, roads, airports, cities, coasts, primary and internal boundaries).
- Displays current date and time.
- Displays parameters of the platform (geographic positions, yaw, pitch, roll, velocity) and its history of the flight trajectory.
- Displays parameters of the radar (carrier frequency, bandwidth, pulse repetition frequency, etc.).
- Displays the border of the scanned region.
- Permits zooming using mouse buttons and a mouse wheel.
- Supplies different kinds of color palettes and shows the color bar.
- Overlays SAR images on vector maps or satellite imagery.

The displaying application should be realized in a way which allows it to be used in field conditions. A small and light laptop, equipped with an Intel Core i7 and 16 GB RAM of memory was chosen. The received data should be calculated to the SAR image and visualized in real-time without freezing the graphics interface.

This paper is an extended version of the paper presented in Signal Processing Symposium, Debe, Poland, June 10 - 12, 2015 [1].

The SARape-project (Synthetic Aperture Radar for all weather penetrating UAV application) was funded by the European Defence Agency (EDA) in the "Defence R&T Joint Investment Programme on Innovative Concepts and Emerging Technologies (ICET)" under the title "Data Capture & Exploitation".

This work was supported by the SARAPE project A- 1089-RT-GC which is coordinated by the European Defence Agency (EDA) and funded by 11 contributing members (Cyprus, France, Germany, Greece, Hungary, Italy, Norway, Poland, Slovakia, Slovenia and Spain) in the framework of the Joint Investment Programme on Innovative Concepts and Emerging Technologies (JIP-ICET).

K. Borowiec is with the Institute of Electronic Systems, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland (e-mail: k.borowiec@elka.pw.edu.pl).

II. DISPLAYING APPLICATION

A. Development Environment

In the development of the SAR imagery displaying application a cross-platform application framework Qt version 4.7 was used [5]. Qt uses standard C++, which is its main advantage. Applications written in the C++ language are characterized by the speed of their execution. This framework provides a complete abstraction of the GUI and uses the native style APIs (Application Programming Interfaces) of different platforms. Therefore, GUI programs created with Qt have a native-looking interface. Qt supports a Meta-Object Compiler (moc) which reads C++ source files, interprets certain macros and uses them to generate added C++ code with meta information about the classes used in the program. This system provides programming features not available natively in C++, such as a dynamic property system and a signal and slots mechanism. The last feature is particularly helpful in communication between objects. An alternative solution, the callbacks technique, can be unintuitive and may suffer from problems in ensuring the type-correctness of callback arguments. Signals and slots are resistant to these disadvantages. A signal is emitted when a particular event occurs in order to notify a different object what happened, and a slot is a function that is called in another object in response to a particular signal. Moreover, the Qt framework has modules essential for solid thread management, communication by network and extensive internationalization support[1].

Additionally, Qt comes with its own set of tools to ease cross-platform development. The first tool is the Qt Creator - a cross-platform integrated development environment for C++ language. This tool is equipped with its own debugger. Another tool is the Qt Designer GUI design functionality.

Outside the raw Qt framework, the QCustomPlot library [6] is also used. The QCP is a Qt C++ widget for plotting and data visualization, focused on offering a high standard of performance for real-time visualization applications.

B. Map Integration

One of the requirements of the SAR imagery displaying application was the capability of showing two types of maps - off- and on-line. The first one was realized using a vector map at level 0 which is an updated and improved version of the National Imagery and Mapping Agency's Digital Chart of the World [7]. The database provides worldwide coverage of vector-based geospatial data which can be viewed at a scale of 1:1.000.000 (i.e. 1cm=10km). The second map was realized using a web mapping service application and technology - Google Maps - provided by Google [8]. Of the many possibilities offered by Google Maps API, street maps and satellite imagery were used. Both services are provided using C++ language combined with JavaScript. In Fig. 1 Google Maps overlaying a vector map is shown.

[1]Qt Linguistic: http://qt-project.org/doc/qt-4.7/linguistmanual. html

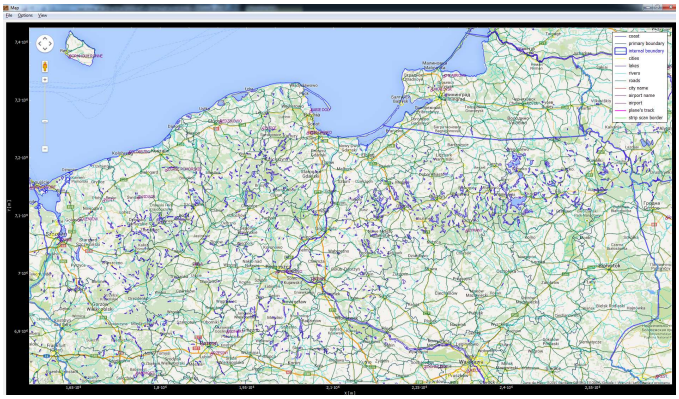

Fig. 1. Google Maps overlaying a vector map

C. Application Functionality

The input data are in the form of UDP packets with consecutive lines of image received from an SAR processer. Because of the two modes of SAR imagery implemented in the application, the GUI was divided into two windows the main window and the map window. Because the SAR processor received the data from two channels, the main window has two symmetrical parts on the left- and right-side corresponding to two channels (see Fig. 2). The main window realizes the simple mode and shows the SAR imagery line by line (in range/cross-range coordinates). In the central part of the window the parameters of the platform and parameters of the radar according to the requirements are displayed.

The map window takes data from one of two channels and shows the SAR imagery, taking into account the current position and flight parameters of the platform. Additionally, in this window it is possible to overlay the SAR image on the vector map or the satellite imagery (see Fig. 3). Moreover, the entire history of the flight trajectory of the platform and the border of the scanned region can be shown after zooming out (see Fig. 4).

The SAR imagery in both modes can be displayed using different kinds of color palettes and transparency levels. Additionally, zooming in and out is available.

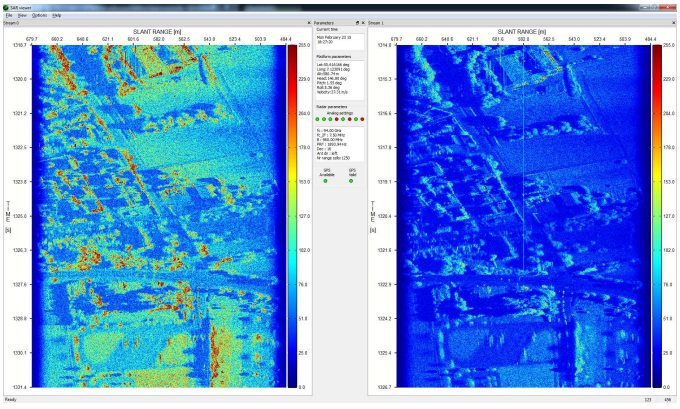

Fig. 2. Main window of the SAR imagery displaying application

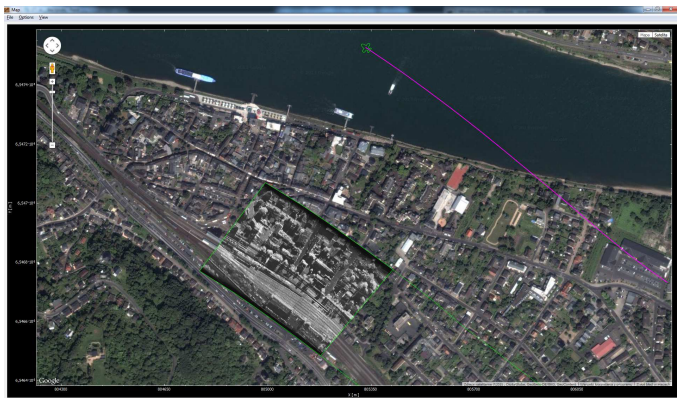

Fig. 3. Map window of the SAR imagery displaying application.

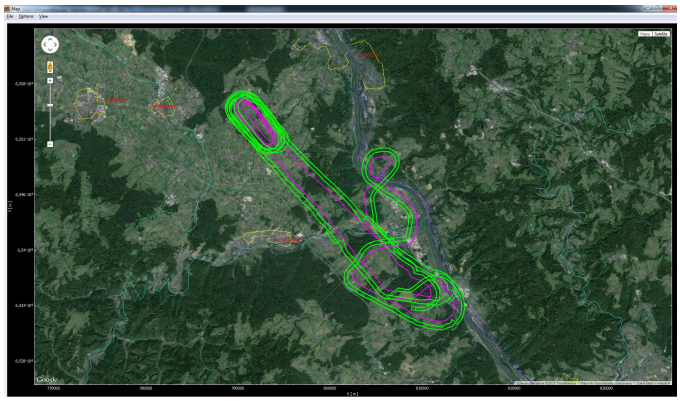

Fig. 4. The history of the flight trajectory and the border of the scanned region

D. Application Architecture

The block diagram of the application is shown in Fig. 5. Threads are working in parallel and communicate with each other using the signals and slots technique. The two incoming streams of data in the form of UDP packets are received in two UdpReceiver threads. The data are stored to circular buffers, and then passed to the SimpleModeDrawer threads, which construct SAR imagery in simple mode. The results are finally forwarded to the main window for display. The UdpReceiver1 has a double role. Firstly, it passes flight and radar parameters to the central part of the main window. Secondly, it commissions the StripScanImage thread to create the SAR imagery, including the flight parameters of the radar platform. In the first step of the algorithm the border of the radar strip scan is calculated. Then the image is filled out using resampling and graphic transformations. The complete image is overlaid on Google Maps using a Mercator projection. Events from user action are served in the main event loop. The rest of the complex and time consuming calculations are carried out in the thread's local event loop. Thereby the application allows the construction of the SAR imagery to be performed in real time.

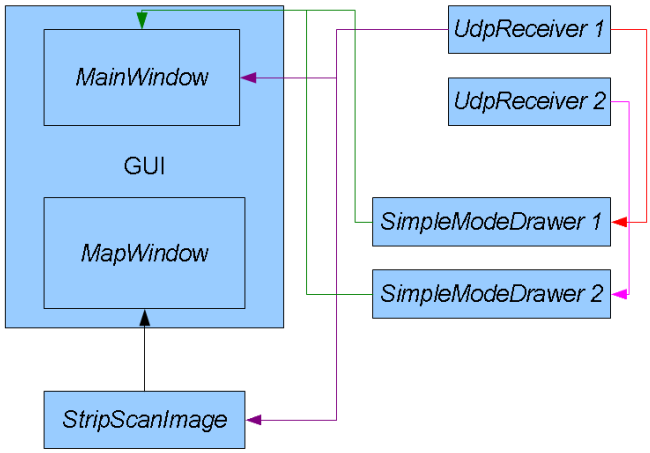

Fig. 5. The block diagram of the application

E. Geographically Oriented SAR Imagery

As the radar operates in FMCW mode, the processing performed to obtain image in the considered case can bo summarized as follows:

- FMCW range compression,
- IMU (Inertial Measurement Unit)-based compensation,
- Signal-based compensation,
- SAR image formation.

All of this steps are out of scope of this paper and are widely described in [4]. As we perform this algorithm on data we obtain 2 dimensional data, where the first dimension (horizontal) are samples from single sounding of FM signal and the second dimension (vertical) are consecutive soundings. Horizontal means the perpendicular plane to the flight path of the radar platform and vertical means parallel plate to the radar platform flight trajectory (Fig. 6).

Signal after SAR processing is expressed in complex numbers, so calculated absolute value represents specific picture element brightness. Every single pixel corresponds to a single resolution cell.

Orientation of the radar platform can be described by three angles: yaw, pitch and roll (Fig. 6). Additionally information about geographic position such as latitude, longitude and altitude allows to find exact position of SAR imagery pixels to their real position on the Earth or proper position on the map.

To calculate size of SAR imagery it is necessary to determine the bounding box of all lines form the single block of data. It can be achieved from placement of the first and last pixel of each line (see Fig. 7). In the following part of this paper major steps leading towards SAR imagery construction are described.

1) Geometric Transformation: Using geometric transformation is necessary to place consecutive lines of SAR imagery at the proper place on the screen. Figure 8 shows the cross section at plane perpendicular to the flight path. The radar operates in 3-dimensional space and the polar coordinates. The scanned swath is placed at some angle to vertical. For this reason, it is necessary to project the every single range cell

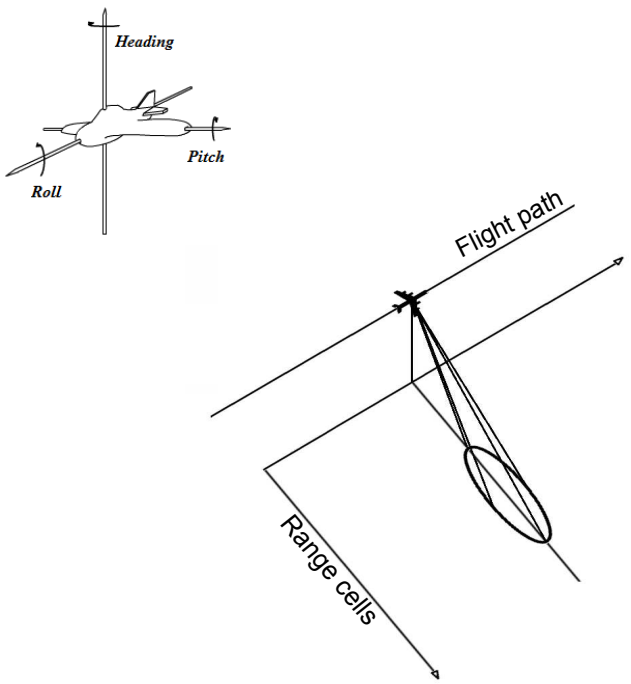

Fig. 6. Platform parameters orientation angles and the idea of SAR imagery

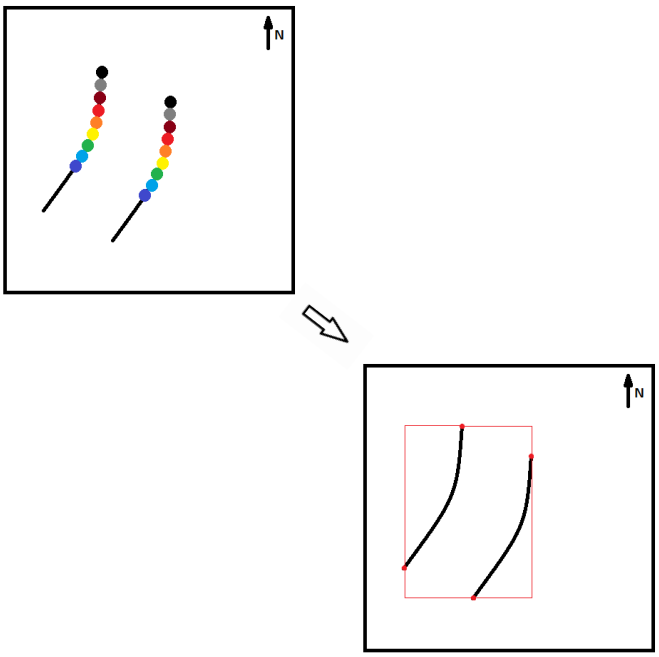

Fig. 7. The first and last range cell's placemensts and the resultant bounding box of SAR imagery lines

from 3-dimensional space to 2-dimensional space. The $|ab|$ line segment placement (4) can be calculated using following equations:

$$|da| = \sqrt{|ca|^2 - |h|^2} \tag{1}$$

$$|cb| = |ca| + nr\_range\_cells \times cell\_size \tag{2}$$

$$|db| = \sqrt{|cb|^2 - |h|^2} \tag{3}$$

$$|ab| = |db| - |da| \tag{4}$$

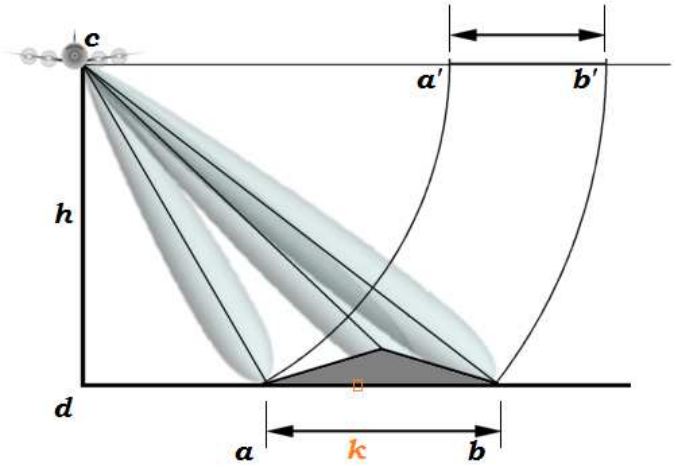

Fig. 8. The SAR imagery geometry

Length of $|ab|$ line segment tells about the scanned region width on the Earth. It also implies directly the single line SAR imagery width in simple mode display. To fill rest of SAR image line, every k range cells position has to be calculated according to the equation:

$$|dk| = |da| + k \times cell\_size. \tag{5}$$

After projection, complete line of SAR imagery is placed on the screen using basic geometric operations. At the Cartesian coordinate system rotation of the point A with coordinates (x_A, y_A) to point A' can be calculated using the following simultaneous equations:

$$\begin{cases} x_{A'} = \cos(\alpha) \cdot x_A + \sin(\alpha) \cdot y_A \\ y_{A'} = -\sin(\alpha) \cdot x_A + \cos(\alpha) \cdot y_A \end{cases}. \tag{6}$$

Translation point A' by vector $\mathbf{T}$ to point A'' expresses simultaneous equations (7).

$$\begin{cases} x_{A''} = x_{A'} + T_x \\ y_{A''} = y_{A'} + T_y \end{cases}, \tag{7}$$

where T_x and T_y are displacement at X and Y coordinates.

Another vital geometric operation is scaling performed according system of equations (8).

$$\begin{cases} x_{A'''} = S_x \cdot x_A \\ y_{A'''} = S_y \cdot y_A \end{cases}, \tag{8}$$

where S_x and S_y is an scaling factor at X and Y coordinates.

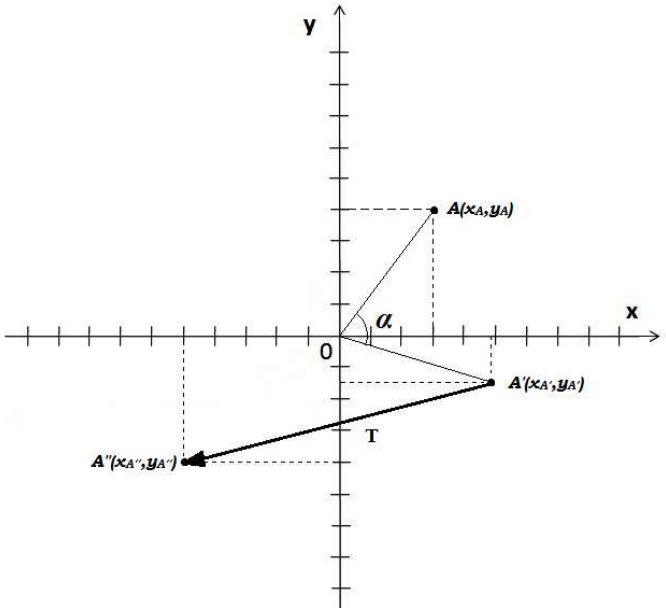

Fig. 9. An α-angle rotation and **T**-vector translation

2) SAR Imagery Resolution: As mentioned above, Synthetic Aperture Radar allows to obtain very high resolution images. Assuming that SAR imagery has 1250 range cells and pulse repetition frequency is equal to 2kHz and the radar platform flies with velocity equal to $40\frac{m}{s}$ number of lines needed to keep aspect ratio defined by equation (9) gives approximately 30.000 lines.

$$\frac{velocity}{PRF} \cdot number\_of\_lines = \frac{1}{2} \cdot 1250[m], \qquad (9)$$

This amount of lines ensures that 1m distance at vertical plane corresponds to 1m distance at horizontal plane. Factor equal to $\frac{1}{2}$ is a consequence of two channels data plotting on the screen at the same time.

Figure 10 shows down-sampling method based on averaging. This method uses two auxiliary matrices. First matrix accumulates values of pixels from original image which will represent single pixel at down-sampled image. In the second matrix counters of how many pixels of the original image will correspond to one pixel in the down-sampled image. In ideal case counter matrix will have the same values at every cell. More complicated case will occur when the radar platform manoeuvres and the flight trajectory is different than straight line.

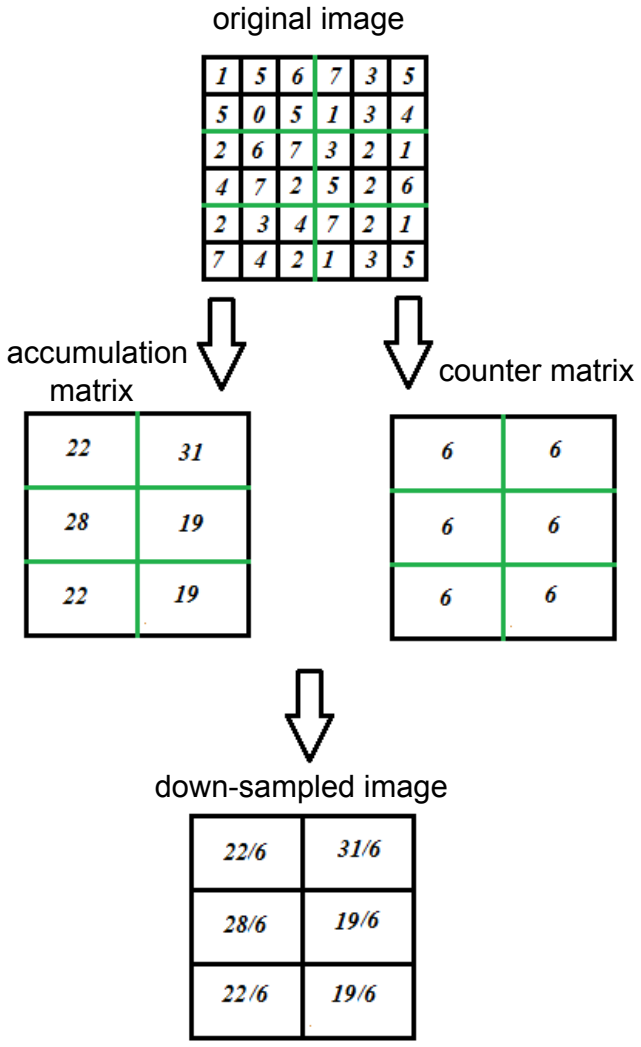

Fig. 10. Downsampling method

Fig. 11. Ultralight aircraft (left) and radar pod (right)

III. RESULTS

All the figures used in this paper are from the results of a measurement campaign carried out for the purpose of verifying the operation of the system realized in the SARape project. The radar pod was assembled under the left wing of an ultralight aircraft (see Fig. 11).

The data were transmitted by the telemetry unit to the ground unit from distances reaching up to 5km. After real-time SAR processing, the high-resolution SAR imagery was shown on the screen. A resolution of up to 15x15cm was obtained. An example of the high resolution SAR imagery is shown in Fig. 12. In the image the lattice towers and power lines are visible.

Another example presented in Fig. 13 shows a characteristic radome of a tracking and imaging radar system with a parabolic dish. The radar is located in Wachtberg, Germany, where the Fraunhofer FHR Institute is located.

The formation of the images in both modes is a computationally extensive task. The application has to process a data stream in the order of several million pixels per second (typically 2.5 million pixels are received for both channels). In

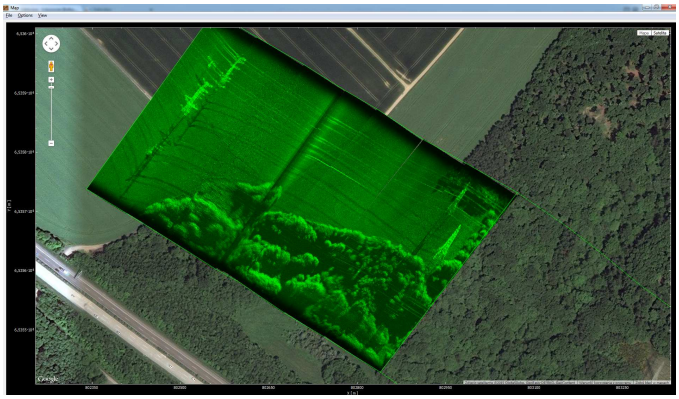

Fig. 12. The radar shadows of the towers and the power lines visible on the SAR imagery

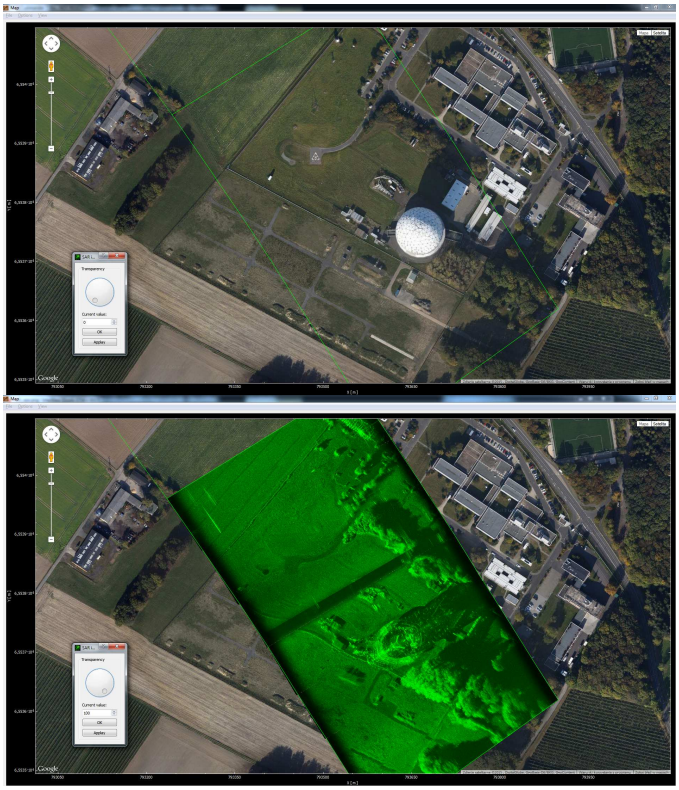

Fig. 13. The Frauhofer FHR in Wachtberg - the SAR imagery overlaying satellite imagery (upper image - 100% level of transparency; lower image - zero level of transparency)

addition, the world coordinate mode requires the performance of complicated geometric transformations in order to properly display the image. Despite these requirements, the application can operate in real time on a laptop. The construction part of the SAR imagery in world coordinate mode takes about 150ms.

IV. Conclusion

In this paper a real-time SAR imagery displaying application is presented. The requirements for the application involved the real-time reconstruction of SAR imagery in two modes, different kinds of color palettes and overlaying on two types of maps. The application performed very well, allowing real-time SAR imagery creation and visualization. The application works efficiently and manages to handle 80 Mbit/s data flow. Results presented in many figures present implementation of assumptions mentioned in the beginning part of this paper. The architecture and the functionality of the application was described too.

The application can be used for the Earth real-time observation for all weather and at any time of the day and night.

References

[1] K. Borowiec, “Real-time synthetic aperture radar imagery displaying application: Implementation and results,” in *Signal Processing Symposium (SPSympo), 2015*, June 2015, pp. 1–4.
[2] M. Caris, S. Stanko, H. Essen, A. Leuther, A. Tessmann, R. Weber, M. Malanowski, P. Samczynski, K. Kulpa, G. Meszoly, C. Topping, G. E. Georgiou, A. C. Papanastasiou, R. Guraly, and Z. Bilacz, “Synthetic aperture radar for all weather penetrating uav application (sarape) - project presentation,” in *Synthetic Aperture Radar, 2012. EUSAR. 9th European Conference on*, April 2012, pp. 290–293.
[3] http://www.project-sarape.eu.
[4] M. Malanowski, G. Krawczyk, P. Samczynski, K. Kulpa, K. Borowiec, and D. Gromek, “Real-time high-resolution sar processor using cuda technology,” in *Radar Symposium (IRS), 2013 14th International*, vol. 2, June 2013, pp. 673–678.
[5] http://qt-project.org/doc/qt-4.7/.
[6] http://qcustomplot.com/.
[7] http://www.mapability.com/index1.html.
[8] https://developers.google.com/maps/.