Radar Data Processing and Visualization on Desktop Platforms

Miloš Jevtić, Milovan Stamatović

***Abstract* — We present an overview of radar data acquisition, processing, and visualization system, implemented mostly in software that runs on a commercially available personal computer. Real-time performance of standard radar data processing procedures and radar data visualization is achieved by utilizing advanced features of modern microprocessors and graphical processors. The use of desktop application programming paradigms enables high system modularity and better interaction between the user and the system.**

***Keywords* — Radar, Plot Extraction, Target Tracking, PPI**

I. Introduction

Radar detects and locates distant objects by transmitting a signal of known waveform and observing the received echoes [1]. Radar systems are in use for seventy years and during that time, they expanded in both capability and in applicability. Applications of modern radar systems include civil air-traffic control, ship safety, space exploration, weather observation, law enforcement and military applications.

Radar data processing and visualization are computationally very demanding operations. In past, highly specialized hardware was required to perform these operations in real-time. Result of this was that the radar data processing and visualization equipment had high development and maintenance costs, was big in size and hard to modify.

In recent years, processing power of commercially available personal computers (PC) has been significantly increased. This is a consequence of both Moore's law and of considerable improvement of system architecture. The result is that a number of computationally demanding tasks, such as 3D animation, video editing, and music production can now be accomplished on a PC.

This trend included radar systems as well. Nowadays, most of the radar data processing procedures and the radar data visualization can be implemented on a commercially available PC. This decreases costs of radar system development and maintenance, while increasing system flexibility since algorithms are implemented as software components that can be modified easily.

We present a radar data acquisition, processing and visualization system. All standard radar data processing procedures and radar data visualization are implemented as software modules that run on a commercially available PC with low-end graphical adapter. Advanced features of modern microprocessors and graphical processors are utilized to achieve real-time performance. In addition, we notice that with a transition to PC-based software implementation many concepts inherent to desktop application programming become available for radar system design. The use of these concepts allows us to bring the interaction between the user and the system to a level that was hard to achieve in traditional radar system implementations.

Miloš Jevtić, Institute Mihajlo Pupin, Volgina 15, 11060 Belgrade, Serbia; (e-mail: milos.jevtic@institutpupin.rs)

Milovan Stamatović, Institute Mihajlo Pupin, Volgina 15, 11060 Belgrade, Serbia; (e-mail: milovan.stamatovic@institutpupin.rs).

II. System Organization

Presented system is designed for use with conventional radar with mechanically rotated antenna. As shown in Fig. 1, the system consists of:

- Data Acquisition Module
- Plot Extractor
- Target Tracker
- Radar Display
- Distribution Subsystem

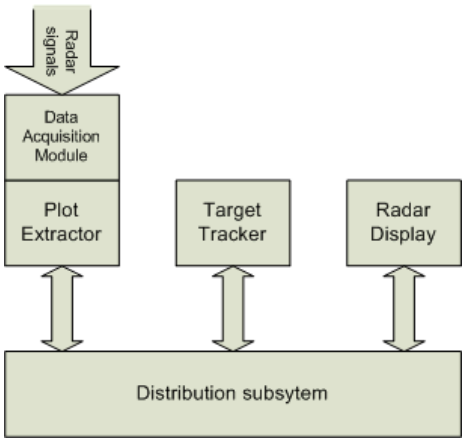

Fig. 1. System organization.

Data Acquisition Module is an interface between the system and the radar. This is a PCI card, which receives analog radar video, video trigger and azimuth. Video is digitized and transferred across the PCI bus using high-speed DMA. Data Acquisition Module is the only component in the system based on specific hardware.

Plot Extractor, Target Tracker and Radar Display are implemented as independent applications/processes, which communicate via Distribution Subsystem.

Distribution Subsystem represents set of libraries developed for local or remote inter-process communication (IPC). It is an application level protocol, built on top of TCP/IP and it is based on a distributed version of publish-subscribe design pattern [2]. This way, weak coupling of software modules is enabled, increasing system modularity, flexibility and stability. Some features of our system that illustrate this are:

- the system can be deployed in a number of configurations (e.g. on a single PC or on multiple networked PCs) without rebuilding the executables,
- software module can be hot-plugged into a running system without interrupting the operation of the system,
- failure in one software module does not affect the rest of the system.

III. Plot Extractor

Plot extractor is software application that analyses radar video and detects potential targets. Processing is based on three main phases: clutter suppression, adaptive thresholding and plot extraction.

Clutter suppression is based on Clutter Map Constant False Alarm Rate (CFAR) algorithm from [3].

Adaptive thresholding is implemented as Cell Averaging CFAR (CA-CFAR) detection, described in [4].

Plot extraction is detection of target-like shapes among video samples that exceed the threshold using standard m-of-n separation criterion.

As shown in Fig. 1, connection between Plot Extractor and acquisition module is direct, which is not in accordance with principle that TCP/IP is our choice for IPC. Since Plot Extractor is the only component in our system that requires high sample rate, i.e. undecimated video signal for processing, it is optimal to place it on the same machine as the signal source. Another consumer of raw video samples is Radar Display. Since Radar Display requires lower sample rates, Plot Extractor also performs downsampling of the video before streaming it on the network.

Plot Extractor, in its basics, is a signal processing application that runs in desktop environment. This implies that we are able to offer user with desktop-like features, to bring user-system interaction on a higher level. For plot extracting process, we find that the classic editing desktop feature could be very useful, so we develop WYSIWYG (What You See Is What You Get) nondestructive extraction parameters editor, shown in Fig. 2.

It is nondestructive because it runs in parallel with the main extractor. When user opens the editor, new instance of extractor is launched. User monitors radar video and if he finds some interesting scenario, for example an interesting mixture of real targets and clutter, he can still the image and make fine tuning of all extraction parameters while being able to see the result of extraction after every change of parameters.

IV. Target Tracker

Plot represents an actual target or false alarm (e.g. clutter, noise, etc.) detected in particular radar scan. In contrast to that, track represents an actual target whose presence is confirmed with a sequence of plots from consecutive radar scans [5]. Each track contains an estimation of the target's kinematic state (position and velocity) and a history of kinematic states from previous scans. Tracks are generated by Target Tracker.

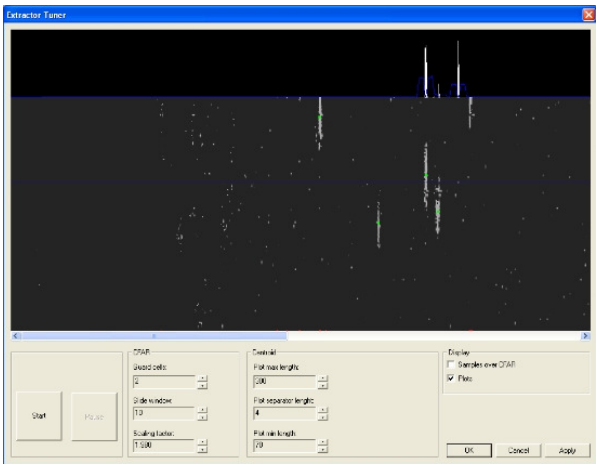

Fig. 2. Extraction parameters editor.

Target Tracker is designed for track-while-scan (TWS) [6] mode of operation, which means that plots are received and processed in regular intervals as the radar regularly scans its search volume. Operation of Target Tracker consists of three main tasks:

1. Data association – Correlation of new plots with already established tracks.
2. Track filtering – Estimating target's kinematic state using associated plot and model of target's motion.
3. Track initiation and maintenance – Detection of the presence of new targets and disappearance of previously present targets.

The goal of data association is to correctly assign plots to existing tracks. The basis for assignment is statistical distance between a given plot and a given track. Statistical distance is calculated as a weighted combination of available plot-to-track coordinate differences.

We choose Global Nearest Neighbor (GNN) [6] for data association method in Target Tracker. In GNN, all possible plot-to-track pairs are taken into account simultaneously. A combination of plot-to-track pairs is sought, such that the sum of plot-to-track statistical distances is minimal. To each track at most one plot is assigned, and each plot is assigned to at most one track. Other data association methods like Joint Probabilistic Data Association (JPDA) [6] and Multiple Hypothesis Tracking (MHT) [6] have much better performance in high false alarm rate (FAR) situations compared to GNN. However, CFAR processing applied in Plot Extractor reduces FAR to such level that GNN association achieves adequate performance, while being computationally less demanding than JPDA and MHT. We implement GNN using Munkres algorithm [7]. Prior to association, we apply ellipsoidal gating technique described in [6] to eliminate unlikely plot-to-track pairs.

When data association is done, track filtering is applied. Track filter operates in predictor-corrector fashion. Each track's kinematic state estimate is predicted based on the model of target motion and the estimate of the kinematic state made on previous scan. Then, track's kinematic state estimate is corrected using plot associated with the track on current scan. Basic tool for track filter implementation is Kalman Filter (KF) [8]. Dynamics of target motion are expressed with transition matrix and process noise covariance matrix of the KF.

Since single KF is inadequate for tracking maneuvering targets, we implement track filter using Interacting Multiple Model (IMM) [9] technique. IMM approach is based on assumption that the target can be in finite number of motion models (e.g. in straight motion or in maneuver). A separate KF is used for each model. Model probability of a particular model is the probability that the target is in that model. Model probabilities for each track are updated from scan to scan, based on plots that are assigned to the track. Each model's KF calculates a separate estimate of target's kinematic state. These estimates are weighted with model probabilities and summed to form an overall estimate of the kinematic state. Before predictions for the next scan are made, new kinematic state estimates are computed for each model via the mixing process.

Plots that are not assigned to existing tracks are used to initiate new tracks. Two consecutive plots are used to create a tentative track. Single KF is used to track a tentative track until enough information is gathered to decide whether the track represents a real target or a sequence of false alarms. If at least M plots were assigned to a tentative track during N consecutive scans, the track is considered a real target and therefore promoted to a confirmed track (which is tracked with IMM thereafter). Otherwise, the track is discarded. When no plots are assigned to a confirmed track in several consecutive scans, the track is deleted.

For debugging, tuning and demonstration purposes, we augment Target Tracker with a graphical user interface (GUI) capable of displaying detailed track histories. Track's detailed history consists of plots that were assigned to the track, position estimates of each model's KF and overall position estimates. In Fig. 3, we give a screenshot showing Target Tracker's GUI with track history of a simulated target. Simulated target is used since real-world datasets containing maneuvering targets are not available to us. The target moves at constant velocity of 300 m/s and makes three constant rate turns with radial accelerations of 10, 20, and 40 m/s^2. Antenna rotation period of 10 s is assumed. White noise with 200 m standard deviation is added to plots of simulated target, modeling position measurement inaccuracy. The target is tracked with IMM consisting of two KF, first for straight motion and second for maneuvers. In Fig. 3, red, green, and black dots connected with lines represent position estimates of straight motion KF, position estimates of maneuver KF and overall position estimates, respectively, while blue dots represent plots.

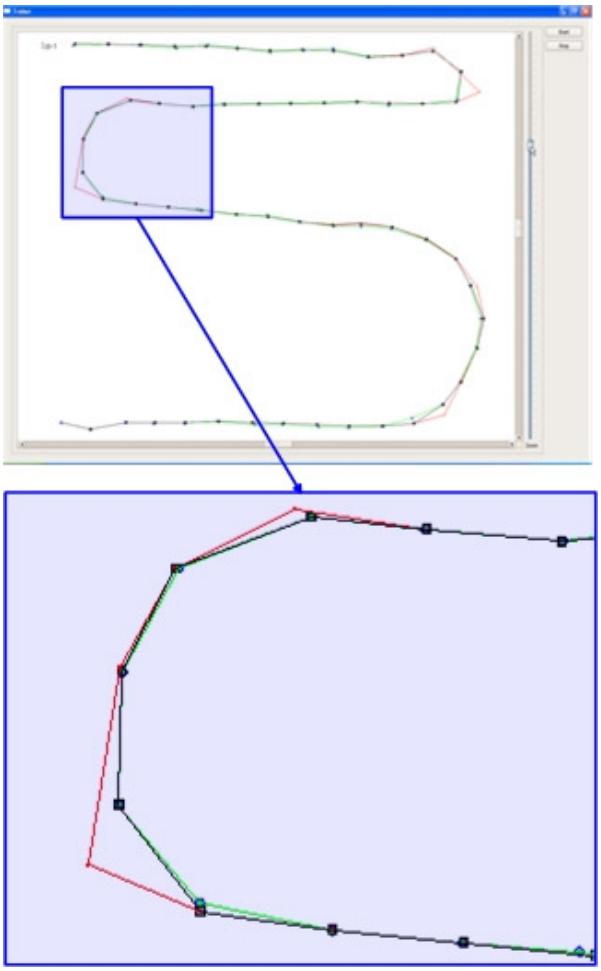

Fig. 3. Target Tracker GUI displaying detailed track history of a simulated target.

V. RADAR DISPLAY

Radar Display is a rich user interface application and it can visualize all kinds of radar data and overlay graphics. It displays: PPI video, A-scan video, plots, tracks and additional overlay graphics such as GIS objects, various markers, polar coordinate system etc.

This application merges modern GUI features and true, dynamic and natural emulation of real analog radar display. From the implementational point of view, the application has to meet the following requirements:

- high resolution of radar picture,
- high refresh/frame rate,
- for every frame all image pixel values must be recomputed,
- a lot of graphical objects must be drawn for every frame.

In achieving these requirements, two features of modern commercially available PCs play significant role. These features are SSE instruction set [10] and graphic acceleration.

SSE stands for Streaming SIMD Extensions, where SIMD is acronym for "Single Instruction Multiple Data", and it is a technique for achieving data level parallelism. SSE technology is intended to speed up those parts of program where the same set of operations is applied to

large number of data points, which is common situation in many multimedia applications.

In Radar Display, SSE instruction set is used for computing image pixel values in order to emulate double persistence effect of analog displays.

Requirement for high frame rate implies that very short time interval is available for drawing additional graphic objects. This is achieved by extensive use of graphic acceleration capabilities of modern PCs.

A screenshot of Radar Display GUI displaying real-world radar data is given in Fig. 4. Light blue squares represent plots, while purple objects represent tracks.

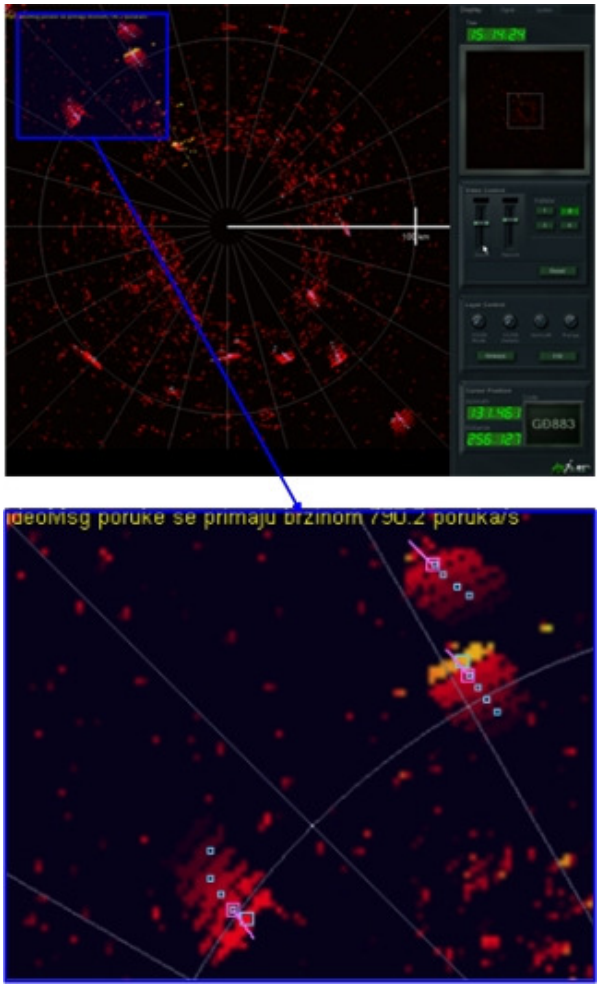

Fig. 4. Radar Display

VI. Applications and Future Work

Primary application area of the presented system are old generation military air-surveillance radar systems. These radar systems either have obsolete data processing subsystems or do not have data processing subsystems at all. In addition, they utilize analog visualization devices, which are hard to maintain. The system we present significantly increases efficiency of old generation radars and prolongs their operational lifetime by introducing modern data processing algorithms and visualization subsystem.

Other possible application areas of the system we present include air-traffic control radars, civil marine radars and vessel tracking service radars.

There are several directions for future development of our system. First, target tracker will be modified to operate on azimuthal sector basis, decreasing the delay between plot detection and track update. Second, support for multiple radar tracking will be added to target tracker. Third, research will be conducted to determine the optimal clutter suppression algorithm.

VII. Conclusion

A system for radar data acquisition, processing and visualization is developed. Most parts of the system are implemented as software components that run on a commercially available PC. We describe how we utilize advanced features of modern microprocessors and graphical processors to achieve real-time performance of standard radar data processing procedures and radar data visualization. Emphasis is put on the fact that the transition from a custom hardware based implementation to a PC-based software implementation enables improved system design. To illustrate this, we describe how the use of desktop application programming paradigms allowed us to achieve high system modularity and better interaction between the user and the system. Possible applications of the system and directions of future development are discussed, too.

References

[1] M. I. Skolnik (Ed.), *Radar Applications*. New York: IEEE Press, 1987.

[2] E. Gamma et al., *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley Professional, 1994.

[3] R. Nitzberg, "Clutter map CFAR analysis," in *IEEE Trans. On Aerospace and Electronic Systems*, Vol. AES-22, Issue 4, pp. 419-421, July 1986.

[4] M. Weiss, "Analysis of some modified cell-averaging CFAR processors in multiple-target situations," in *IEEE Trans. On Aerospace and Electronic Systems*, Vol. AES-18, No. 1, pp. 102-114, January 1982.

[5] W. G. Bath, G. V. Trunk, "Automatic detection, tracking, and sensor integration," in *Radar Handbook*, 3rd ed., M. I. Skolnik, Ed. New York: McGraw-Hill, 2008.

[6] S. Blackman, R. Popoli, *Design and Analysis of Modern Tracking Systems*. Norwood, MA: Artech House, 1999.

[7] S. Blackman, *Multiple Target Tracking with Radar Applications.* Norwood, MA: Artech House, 1986.

[8] R. E. Kalman, "A new approach to linear filtering and prediction problems," in *Trans. of the ASME-Journal of Basic Engineering*, 82 (Series D), pp. 35-45, 1960.

[9] Y. Bar-Shalom (Ed.), *Multitarget-Multisensor Tracking: Applications and Advances*, Vol II. Norwood, MA: Artech House, 1992.

[10] J. Abel et al., "Applications Tuning for Streaming SIMD Extensions," in *Intel Technology Journal*, Q2, 1999.