순서통계에 근거한 개선된 CFAR 검파기의 하드웨어 구조 제안

현유진* · 이종훈**

Advanced OS-CFAR Processor Design with Low Computational Effort

Eugin Hyun* · Jong-Hun Lee**

본 연구는 교육과학기술부에서 지원하는 대구경북과학기술원 기관고유사업에 의해 수행되었습니다. (11-RS-02)

요 약

순서통계에 근거한 CFAR(Constant False Alarm) 검파기(이하 OS-CFAR)는 다중 타깃(Target) 환경의 차량용 레이더에 아주 유용 사용되는 알고리즘이다. 그러나 정렬 알고리즘을 사용하기 때문에 일반적인 셀-평균 CFAR 검파기(이하 CA-CFAR)에 비해 계산량이 많아 실시간 구현에 어려운 점이 있다. 본 논문에서는 보다 낮은 계산량을 가지는 OS-CFAR 구조를 제안하였다. 제안된 방법에서는 정렬 알고리즘이 단 한번 만 수행되기 때문에 이를 통해 많은 계산량을 줄일 수 있다. 특히 고속 정렬 알고리즘을 사용하는 경우 통상적인 OS-CFAR 구조와 비교하여 데이터양에 상관없이 항상 계산속도가 빠름을 확인 할 수 있다. 또한 본 논문에서는 실제 레이더 수신 데이터를 이용하여 제안된 방법에 적용한 결과도 제시하였다.

ABSTRACT

An OS-CFAR (Ordered Statistics CFAR) based on a sorting algorithm is useful for automotive radar systems in a multi-target situation. However, while the typical cell-averaging CFAR has low computational complexity, the OS-CFAR has much higher computation effort. In this paper, we design the new OS-CFAR architecture with a low computational effort. In the proposed method, since one time sorting processing is performed for the decision of the CFAR threshold, the whole processing effort can be reduced. When the fast sorting technique is employed, the computing time of the proposed OS-CFAR is always much shorter compared with typical OS-CFAR method regardless of the data size. We also present the processing result of proposed architecture using the real radar data.

키워드

CFAR 검파기, 순서통계, 레이더

Key word

OS-CFAR, CFAR, Radar

* 정회원 : 대구경북과학기술원 (braham@dgist.ac.kr)
** 정회원 : 대구경북과학기술원 (교신저자)

접수일자 : 2011. 10. 17
심사완료일자 : 2011. 11. 14

Ⅰ. 서 론

레이더 기반의 차량용 능동 안전 시스템은 최근 지능형 자동차의 핵심 키워드로 부각되고 있다. 레이더는 카메라, 초음파, 레이저 등과 달리 외부 환경에 보다 덜 민감하고 주야 및 날씨에 상관없이 원하는 정보를 획득할 수 있기 때문에 전방 및 측,후방 차량 탐지를 위해 널리 활용되고 있다[1].

레이더는 송신 신호가 타깃으로부터 반사된 수신 신호의 시간 지연과 도플러 천이 값을 탐지하여 거리 또는 속도를 측정한다. 이때 이상적인 경우에는 수신 신호의 세기가 특정 기준치(Threshold)보다 클 경우 타깃으로 탐지하고 그 이하일 경우 노이즈로 판단한다[2-3].

레이더의 성능지표는 탐지율과 오경보율(False alarm rate)로 나타낼 수 있다. 잡음 분포가 수시로 변동하는 실제 도로 환경에서는 다수의 클러터(Clutter)가 존재하게 되므로 고정된 특정 기준치를 이용하여 타깃을 탐지하면 잡음 수준에 따라 오경보율이 변동한다. 차량용 레이더의 경우 비록 탐지 확률이 높더라도 오경보율이 수시로 변동한다면 신뢰성 있는 타깃 탐지 정보를 검출 할 수 없다. 따라서 고정된 오경보율을 유지하기 위해 기준치 값이 가변 하는 CFAR 검출기가 사용된다[2-3].

가장 일반적으로 사용되는 CFAR 검출기는 셀-평균(Cell average) CAFR 검출기로 동작 과정이 그림 1에 나타내었다.

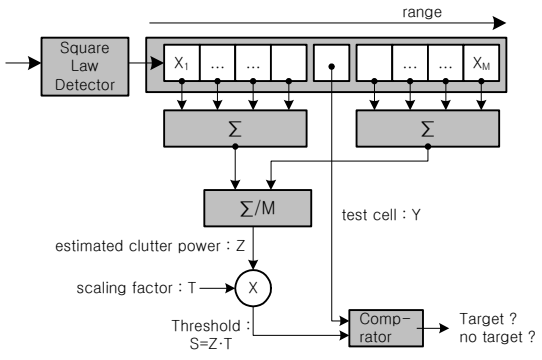

그림 1. 셀-평균 CFAR 검출기 블록다이어그램
Fig. 1 The Block-diagram of CA-CFAR detector

CA-CFAR 알고리즘은 슬라이딩 윈도우와 셀(Cell)을 기반으로 하여 동작을 한다. 슬라이딩 윈도우 내부에는 M개의 참조-셀(Reference-cell)로 이루어져 있고, 이들의 평균값을 구함으로써 클러터-전력(Clutter power), Z를 획득한다. 스케일링-인자(Scaling-factor)와 곱셈 연산을 통해 기준치, S가 구해진다. 마지막으로 테스트-셀(Test-cell), Y와 기준치, S를 비교하여 타깃 여부를 결정한다. 이런 연산은 모든 거리 데이터에 적용된다.

CA-CFAR 탐지기는 간단한 구조와 낮은 계산량으로 인해 가장 일반적으로 사용되는 알고리즘이다. 그러나 CA-CFAR 알고리즘은 다중 타깃 환경과 비균등(Non-uniform) 클러터 환경에서는 성능이 떨어지는 단점이 있다[2-5].

이를 해결하기 위해 제안된 알고리즘이 바로 OS-CFAR(Ordered Statistics CFAR)으로, 이 알고리즘은 다양한 클러터 환경에서 다중 타깃을 탐지해야 되는 차량용 레이더에 적합하다고 알려져 있다[6]. 하지만 이 알고리즘은 CA-CFAR에 비해 계산량이 복잡하다는 단점이 있다[4-6].

이에 선행 연구에서는 보다 계산량을 향상시킨 OS-CFAR 탐지기를 제안하였다[7]. 본 논문에서는 제안된 OS-CFAR 탐지기의 동작과정을 보다 자세히 설명하고, 이를 효과적으로 지원하기 위한 하드웨어 구조를 제안한다. 또한 소모되는 하드웨어를 비교하며, 시뮬레이션을 통해 동작 성능이 동일함을 확인한다.

본 논문의 구성은 Ⅱ장에서 OS-CFAR 알고리즘을 소개하고, Ⅲ 제안된 OS-CFAR 및 하드웨어 아키텍처를 제시한다. Ⅳ장에서 시뮬레이션 결과를 소개하고, Ⅴ장에서 결론을 맺고자 한다.

Ⅱ. OS-CFAR 탐지기의 개요

그림 2는 일반적인 OS-CFAR 탐지기의 블록다이어그램을 나타낸다.

먼저 수신된 거리-데이터는 자승-검파기(Square-law detector)를 통해 셀-레지스터(Cell-register)에 저장된다. 저장된 데이터는 슬라이딩 윈도우에 의해 M개 참조-셀과 테스트-셀이 결정된다. M개 참조-셀은 크기 값이 큰 순서대로 정렬(Sorting)된 후 k번째 데이터가 클러터-전력, Z로 선택된다. CA-CFAR와 마찬가지로 스케일링-인

자와의 곱셈 연산을 통해 기준치, S가 구해지고, 이를 이용해 테스트 셀 Y의 타깃 여부가 결정된다.

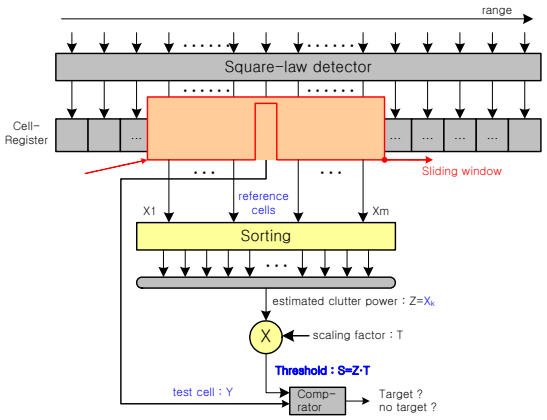

그림 2. 일반적인 OS-CFAR 검출기
Fig. 2 Typical OS-CFAR detector

OS-CFAR 탐지기가 CA-CFAR와 비교하여 계산량이 많은 이유는, 바로 정렬 알고리즘 때문이다. 버블-정렬(Bubble-sorting)과 같은 가장 간단한 정렬 알고리즘을 사용하는 경우의 계산 복잡도는 평균 $M(M-1)/2$이다. 또한 퀵-정렬(Quick-sorting) 방법과 같은 고속 정렬 알고리즘의 경우 계산 복잡도는 평균 $M \cdot \ln M$이다. 여기서 M은 윈도우 크기로 한 번에 처리하는 테스트-셀의 개수와 동일하다.

또한 전체 거리 데이터의 개수가 N개라면, 비교해야 할 테스트-셀이 N개이므로 정렬 알고리즘은 N번 반복된다.

만약 스케일링-인자를 곱하는 연산과 비교 연산이 각각 1 클럭이 소모된다고 가정하면, 버블-정렬과 퀵-정렬을 사용한 OS-CFAR 알고리즘의 복잡도는 각각 수식 (1)과 (2)와 같다.

$$S_1(N) = N\left(\frac{M(M-1)}{2}+2\right) \quad (1)$$

$$S_2(N) = N(M\ln M+2) \quad (2)$$

따라서 OS-CFAR의 계산량을 향상시키기 위해서는 바로 정렬 알고리즘의 동작 횟수를 줄이는 방법이 필요하다.

Ⅲ. 제안된 OS-CFAR 탐지기

앞서 설명하였듯이, OS-CFAR 탐지기는 정렬 알고리즘이 N번의 거리-데이터만큼 반복되면서 전체 계산량을 증가시킨다.

이를 해결하기 위해 그림 3과 같이 OS-CFAR 알고리즘을 제안한다.

기본적인 CFAR 구조와 마찬가지로, 먼저 수신된 데이터는 자승-검파기를 통해 셀-레지스터에 저장된다. 저장된 데이터는 모두 정렬 알고리즘을 통해 크기 값이 큰 순서대로 정렬된 후 정렬-레지스터(Sorting-register)에 저장된다. 이때 크기 값 뿐 아니라, 원래 셀-레지스터에 저장되어 있든 레지스터 번호까지도 같이 저장된다.

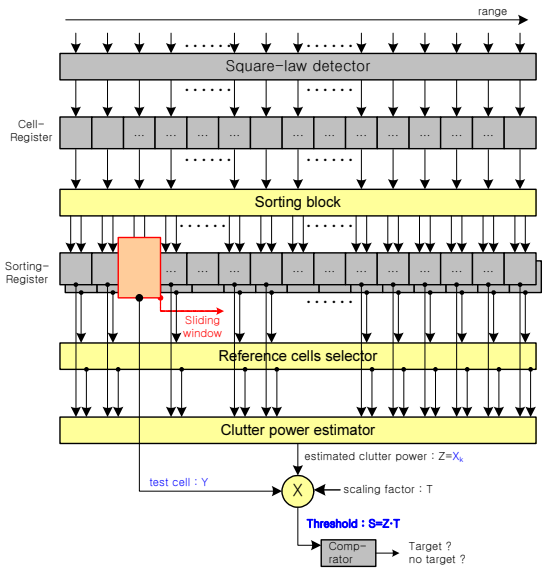

그림 3. 제안된 OS-CFAR 검출기
Fig. 3 The proposed OS-CFAR detector

정렬-레지스터에 저장된 데이터는 참조-셀-선택기(Reference cell selector)에 의해, 테스트 셀에 해당하는 참조 셀들이 선택된다.

선택된 참조 셀의 정보를 이용하여 클러터-전력-검출기(Clutter power estimator)에 의해 k번째 데이터가 클러터-전력, Z로 선택된다. 그 이후 스케일링-인자와의 곱셈 연산을 통해 기준치, S가 구해지고, 이를 이용해 테스트 셀 Y의 타깃 여부가 결정된다.

참조-셀-선택기와 클러터-전력-추정기는 정렬-레지스터에 저장된 데이터에 모두 적용되므로 모두 N번 동작한다.

그림 4는 설계된 정렬-레지스터의 구조이다. 셀-레지스터에 저장된 데이터의 크기를 이용하여 $X_{(1)} > X_{(2)} > \cdots > X_{(N)}$로 정렬 한 후 이를 저장한다. 이때 $X_{(1)},\cdots,X_{(N)}$의 데이터들이 원래 저장되어 있든 셀-레지스터 번호, $\&X_{(1)},\cdots,\&X_{(N)}$도 함께 저장한다. 보다 구체적인 예를 들어, 셀-레지스터에 저장된 데이터 R_7이 내림차순 정렬 후 $X_{(3)}$에 해당한다면, $X_{(3)}$은 R_7이 되고 $\&X_{(3)}$은 셀-레지스터 R_7에 해당하는 7이 된다.

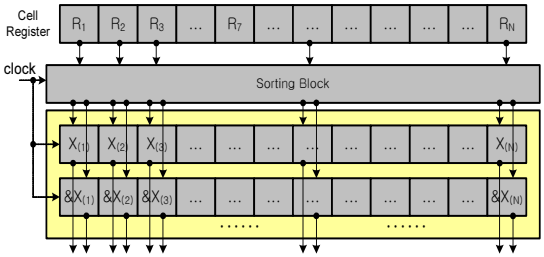

그림 4. 설계된 정렬-레지스터
Fig. 4 The designed sorting-register

정렬 알고리즘이 수행 시 동작 소모시간은 버블-정렬일 경우 평균 $N(N-1)/2^2$ 클럭, 퀵-정렬 평균 $M\cdot\ln M$ 클럭 정도가 소모될 것이다. 일단 한번 정렬 후 정렬 레지스터에 저장이 되고나면, 새로운 거리-데이터가 수신될 때 까지는 정렬 알고리즘은 동작 되지 않는다.

그림 5는 참조-셀-선택기의 하드웨어 구조를 나타낸다. 정렬-레지스터에 저장된 $\&X_{(1)},\cdots,\&X_{(N)}$값은 컨트롤-블록(Control block)에 의해, 차례대로 테스트 셀에 해당하는 $\&Y$로 선택된다. 이는 총 N번 반복되고, 이는 마치 기존 CFAR 탐지기의 슬라이딩 윈도우 기법과 유사하게 동작한다.

선택된 $\&Y$과 $\&X_{(1)},\cdots,\&X_{(N)}$의 값은 비교기를 통해 $B_{(1)},\cdots,B_{(N)}$값으로 출력되는데, 이 값이 '1'이면 $X_{(n)}$들이 테스트셀 Y의 참조셀임을 나타낸다.

여기서 $n=1\sim N$이다 예를 들어, $\&Y$이 7이고 참조-셀을 나타내기 위한 윈도우 크기가 8이라면, $\&X_{(n)}$의 값이 3~6, 8~11에 해당하는 $B_{(n)}$만 '1'이 되고 나머지는 모두 0으로 나타난다. 여기서 $n=1\sim N$에 해당한다.

따라서 비교기의 수식은 아래와 같고, $B_{(1)},\cdots,B_{(N)}$으로 구성된 출력값은 N비트 스트링에 해당한다.

- if [&X(n) ≥ &Y-(M/2)] & [&X(n) ≤ &Y+(M/2)] & [&X(n) ≠ &Y] then 1
- else 0, where n=1~N

계산에 필요한 클럭 수는 $B_{(1)},\cdots,B_{(N)}$이 레지스터에 저장되는 1 클럭이면 가능하고, 이는 모두 N번 수행 될 것이다.

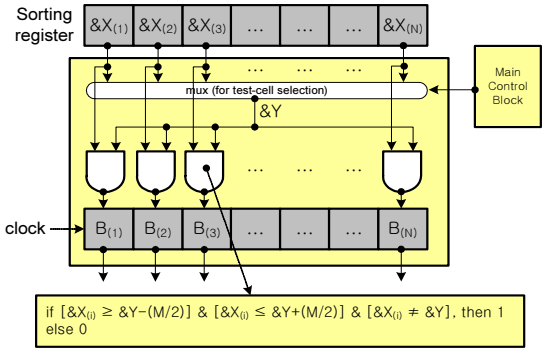

그림 5. 설계된 참조-셀-선택기
Fig. 5 The designed reference cells selector

다음은 클러터-전력-검출기의 하드웨어 구성도이다. 참조-셀-선택기로부터 출력된 N비트 스트링에서 k번째 '1'을 찾은 후, 이에 해당하는 $X_{(n)}$의 값을 찾게 된다.

이때 k번째 '1'을 찾는 과정은 트리 구조로 수행되므로 계산량은 lnN 클럭 정도 소모된다. 스케일링-인자를 곱하는 연산과 비교 연산이 각각 1 클럭이 소모된다고 가정하면, 제안된 OS-CFAR 알고리즘의 최종 계산량은 수식 (3) 및 (4)와 같이 표현 할 수 있다. 여기서 S_3는 버블-정렬을 사용한 경우이고 S_4는 퀵-정렬을 사용한 경우에 해당한다.

$$S_3(N) = \frac{N(N+1)}{2} + N(3+\ln N) \quad (1)$$

$$S_4(N) = N\ln N + N(3+\ln N) \quad (2)$$

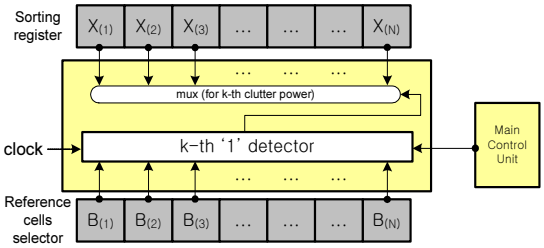

그림 6. 클러터 전력 검출기
Fig. 6 The designed clutter power estimator

Ⅵ. 시뮬레이션 결과

일반적인 OS-CFAR와 제안된 OS-CFAR의 계산 시간을 분석하기 위해 데이터의 크기를 증가시키며 시뮬레이션 하였다. 이때 윈도우 크기는 일반적인 CFAR에서 통상적으로 사용하는 8, 16, 32로 선택하였다.

그림 7은 버블-정렬 알고리즘을 이용한 경우이다. 일반적인 방법의 윈도우 크기가 8일 때 데이터가 약 40개를 넘는 경우에는, 제안된 방법의 계산시간이 더 많이 소모됨을 알 수 있다. 마찬가지로 윈도우 크기가 16일 및 32일 때는, 전체 데이터 크기가 약 220개와 960개 이상인 경우에는 제안된 방법의 소모 클럭이 더 많음을 알 수 있다. 그 이유는 일반적인 방법의 경우에는 정렬 알고리즘에 적용되는 데이터의 개수가 윈도우 크기로 아주 작지만, 제안된 방법의 경우에는 윈도우 크기와 상관없이 모든 데이터에 정렬 알고리즘이 적용되기 때문이다.

그림 8은 퀵-정렬 알고리즘을 이용한 경우의 계산 소모시간을 비교한 것이다. 이 경우 일반적인 방법의 윈도우 크기와 상관없이 제안된 방법의 계산시간이 훨씬 줄어듦을 알 수 있다. 다음은 일반적인 방법과 제안된 방법에서 소모되는 하드웨어 리소스의 주요 로직 개수를 비교한 결과를 표 1에 나타내었다. 여기서 자승-검파기는 제외하였다. 각 셀의 데이터 비트크기는 L, 데이터 개수는 N, 윈도우 크기는 M이다. 또한 퀵-정렬 알고리즘이 버블-알고리즘에 비해 몇 개의 스택이 추가적으로 필요하지만 본 논문에서는 이를 고려하지 않았다.

일단 셀-레지스터, 곱셈기, 비교기는 일반적인 방법과 제안된 방법에 모두 공통으로 사용된다. 일반적인 방법의 경우 윈도우 내의 데이터 M개를 정렬 후 저장을 해야 하므로, 추가적으로 L비트 레지스터 M개가 필요하다.

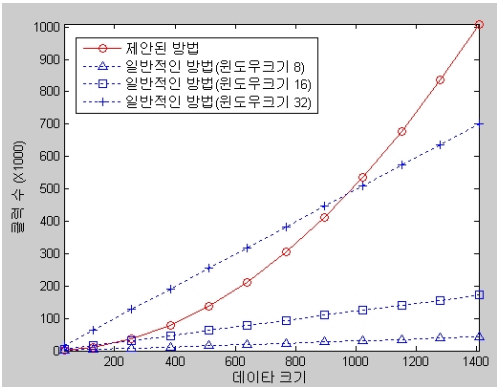

그림 7. 버블-정렬 알고리즘을 이용한 경우의 OS-CFAR 계산량 비교
Fig. 7 Comparison of computation time in OS-CFAR processor with bubble sorting algorithm

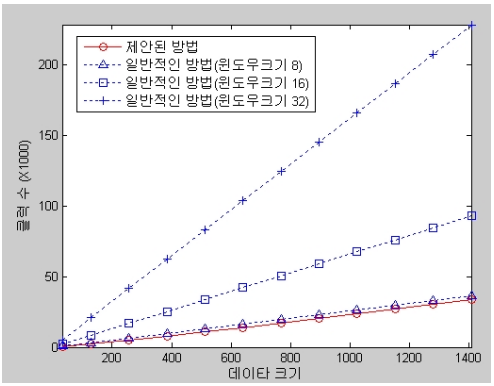

그림 8. 퀵-정렬 알고리즘을 이용한 경우의 OS-CFAR 계산량 비교
Fig. 8 Comparison of computation time in OS-CFAR processor with quick sorting algorithm

또한 클러터-전력을 선택하기 위해 L비트 입력 M개의 멀티프렉서가 필요하다. 제안된 방법에서는 정렬 결과를 그대로 셀-레지스터에 다시 저장을 하면 되므로, 추가적인 레지스터는 필요 없다. 다만 정렬 알고리즘을 수행 후, 그림 4와 같이 셀-레지스터의 번호도 함께 저장을 해야 하므로, $\log_2 N$ 비트의 N개 레지스터가 추가적으로 필요하다. 또한 그림 5의 참조-셀 선택기에서는 $\log_2 N$ 비트 비교기 N개와 1비트 레지스터 N개가 추가로 필요하다. 마지막으로 그림 6의 클러터-전력 검출기에서는 1의 개수를 카운터 하여, k번째 1을 찾아야 하므로, 1비트 N개 입력 덧셈기가 필요하다.

표 1. 일반적인 방법과 제안된 방법을 지원하기 위한 하드웨어 리소스
Table. 1 Hardware resource to support typical and proposed method

	일반적인 방법	제안된 방법
Cell Register	L비트 레지스터 N개	L비트 레지스터 N개
Sorting Algorithm	L비트 레지스터 M개	$\log_2 N$ 비트 레지스터 N개
Reference Cells Selector	-	① $\log_2 N$ 비트 비교기 N개 ② 1 비트 레지스터 N개
Clutter Power Estimator	L비트 M to 1 멀티플렉서	1비트 N개 입력 덧셈기 1개
Multiplier	L비트 2입력 곱셈기 1개	L비트 2입력 곱셈기 1개
Comparator	L비트 2입력 비교기 1개	L비트 2입력 비교기 1개

위의 계산을 바탕으로 사용된 레지스터의 개수를 비교한 결과를 그림 9에 나타내었다. 이때 데이터의 크기는 32비트로 가정하였으며, 데이터의 개수는 32부터 1400개까지 증가시켰다. 그 결과 일반적인 방법에서 필요한 레지스터는 1비트 기준으로 약 46,000개가 필요하고, 제안된 방법의 경우 약 61,200개가 필요함을 알 수 있다. 데이터의 개수가 늘어나는 경우 필요한 하드웨어 리소스는 더욱 늘어날 것으로 보인다.

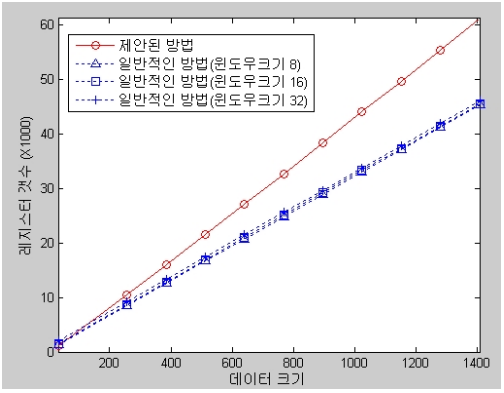

그림 9. OS-CFAR 하드웨어 리소스 소모량 비교
Fig. 9 Comparison of required hardware resource in OS-CFAR processor

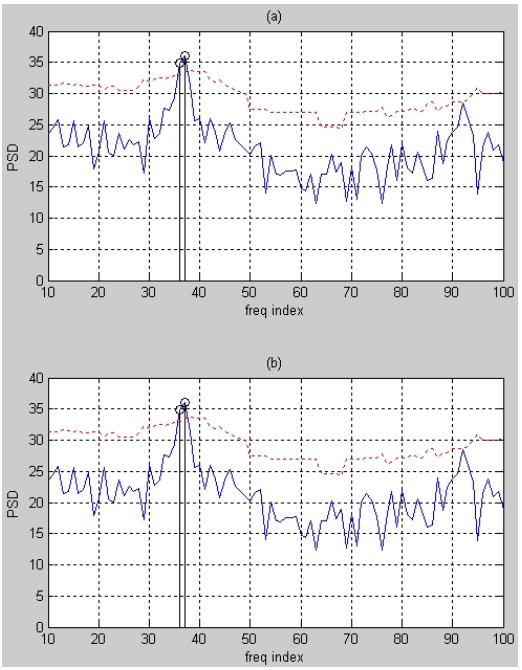

그림 10. 실제 레이더 수신신호에 적용한 결과
Fig. 10 The processing result using the real data received from the radar

위의 그림 10은 실제 77GHz FMCW (Frequency Modulation Continuos Wave) 레이더를 이용하여 단일 이동타깃에 대해 적용한 결과이다. 가로축은 주파수 인덱스를 나타내며, 세로축은 전력밀도의 dB 스케일을 나타낸다. 그림 10(a)은 일반적인 OS-CFAR를 적용한 경우이며, 그림 10(b)는 제안된 OS-CFAR를 적용한 경우이다. 탐지된 모든 시간에 대한 비교 결과 역시 두 개의 알고리즘 역시 동일한 성능이 나옴을 확인 할 수 있었다.

V. 결론

본 논문에서는 일반적인 OS-CFAR에 비해 계산량을 향상시킨 프로세서 구조를 제안하였다. 제안된 방법에서는 정렬 알고리즘을 한번만 수행함으로써 전체 동작 시간을 줄일 수 있다. 제안된 방법은 시뮬레이션을 통해 일반적인 OS-CFAR와의 계산량 차이를 분석하였으며,

또한 실제 소모되는 하드웨어 리소스도 비교 분석하였다. 마지막으로 실제 레이더 수신 데이터를 이용하여 데이터 처리된 결과를 비교하였다. 제안된 아키텍처는 향후 레이더 신호처리 알고리즘을 구현함에 있어서 효과적으로 활용 될 수 있을 것이다.

참고문헌

[1] 김상동, 현유진, 이종훈, 최준혁, 박정호, 박상현, “차량용 FMCW 레이더의 탐지 성능 분석 및 신호처리부 개발”, 한국해양정보통신학회논문지 제14권 12호, pp. 2629-2635. 2010년 12월

[2] Rohling H., : ‘Radar CFAR Thresholding in Clutter and Multiple Target Situations’, IEEE Transaction on Aerospace and Electronics System, Vol. AES-19, No. 4, pp. 608-620, July, 1983

[3] 송우영, 노수현, 정철호, 곽영길, “다중 셀 평균 기반 CFAR 검출을 이용한 SAR 영상 표적 탐지 기법”, 한국전자파학회론문지 제21권 제2호, pp. 164-169, 2010년 2월

[4] Longo M., Lops M., ″OS-CFAR Thresholding in Decentralized Radar Systems″, IEEE Transaction on Aerospace and Electronics Systems, Vol. 32, Issue 4, pp. 1257-1267, August, 2006

[5] Mark A. Richards : ‘Fundamentals of Radar Signal Processing’, MCGraw-Hill, pp. 347-383

[6] Stephen Blake : ‘OS-CFAR Theory for Multiple Targets and Nonuniform Clutter’ , IEEE Transaction on Aerospace and Electronics System, Vol. 24, No. 6, pp. 785-790, Novemver. 1988

[7] Eugin Hyun, Jong-Hun Lee, ″A New OS-CFAR Detector Design″, First ACIS/JNU International Conference on Computers, Networks, Systems, and Industrial Engineering, pp. 133-136, May, 2011

저자소개

현유진(Eugin Hyun)

1999년, 영남대학교 전자공학과 공학사
2001년, 영남대학교 전자공학과 공학석사
2005년, 영남대학교 전자공학과 공학박사
2005년～현재, 대구경북과학기술원(DGIST) 로봇시스템연구부 선임연구원
2008년～현재, 영남대학교 전자공학과 겸임교수
※관심분야 : 레이더 디지털 신호처리, 디지털 신호처리 프로세서 구현, 레이더 체계 공학

이종훈(Jong-Hun Lee)

1996년, 성균관대학교 전자공학과 공학사
1998년, 성균관대학교 전기전자 컴퓨터공학과 공학석사
2002년, 성균관대학교 전기전자 컴퓨터공학과 공학박사
2002년～2005년, 삼성전자 통신연구소 책임연구원
2005년～현재, 대구경북과학기술원(DGIST) 로봇시스템연구부 선임연구원(과제책임자)
※관심분야 : 레이더 신호처리, FMCW/UWB 레이더, 레이더 체계공학