(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2016-0026110
(43) 공개일자 2016년03월09일

(51) 국제특허분류(Int. Cl.)
A61B 8/14 (2006.01)
(21) 출원번호 10-2014-0114032
(22) 출원일자 2014년08월29일
심사청구일자 2014년08월29일

(71) 출원인
서강대학교산학협력단
서울특별시 마포구 백범로 35 (신수동, 서강대학교)
(72) 발명자
송태경
서울특별시 종로구 평창문화로 156 101동 703호 (평창동, 평창동 롯데캐슬 로잔)
공우규
서울특별시 도봉구 해등로 242-11 103동 304호(쌍문동, 성원아파트)
(74) 대리인
특허법인충현

전체 청구항 수 : 총 11 항

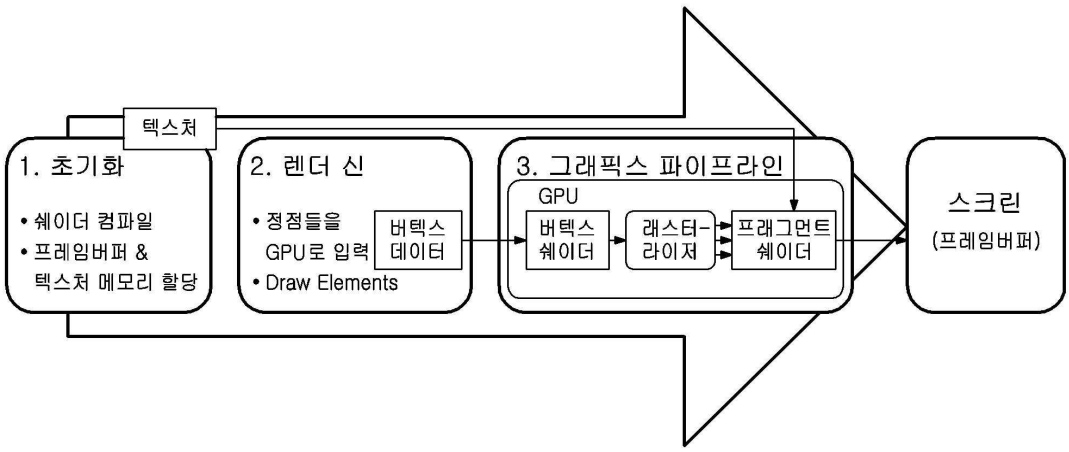
(54) 발명의 명칭 **스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법**

(57) 요 약

본 발명은 스마트 기기가 초음파 신호를 입력받아, 렌더 사이클(Render Cycle)을 통해 처리하여 초음파 영상을 생성하는데 이용되는 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법에 관한 것으로, 제1 렌더 사이클을 통해 빔포밍된 초음파 신호를 입력받아, 상기 초음파 신호로부터 DC 성분을 제거한 후, DC 성분이 제거된 초

(뒷면에 계속)

대 표 도 - 도1



음파 신호로부터 in-phase 성분과 quadrature 성분을 나누어 각각 출력하는 단계; 제2 렌더 사이클을 통해 상기 in-phase 성분과 quadrature 성분을 갖는 초음파 신호에 대하여 quadrature 복조 및 데시메이션(decimation) 처리를 수행하는 단계; 제3 렌더 사이클을 통해 블랙홀로 간주하는 문턱치값을 입력받아, 상기 문턱치값과 상기 제2 렌더 사이클로부터 입력받은 초음파 신호의 크기를 상호 비교하여 상기 초음파 신호에 대한 블랙홀을 제거하는 단계; 제4 렌더 사이클을 통해 상기 블랙홀이 제거된 초음파 신호에 대한 에지 강화를 수행하는 단계; 및 제5 렌더 사이클을 통해 상기 에지 강화가 수행된 초음파 신호에 대하여 스캔 변환을 수행하는 단계;를 포함하되, 상기 제1 내지 제5 렌더 사이클은 버텍스 쉐이더(vertex shader) 단계, 래스터라이저(Rasterizer) 단계 및 프래그먼트 쉐이더(fragment shader) 단계를 포함하는 그래픽스 파이프라인(Graphics pipeline) 구조로 이루어지며, 상기 스마트 기기에서 구동되는 모바일 GPU(Graphic Processing Unit)가 상기 프래그먼트 쉐이더 단계에서 수행되도록 할당된 연산과정의 일부를 상기 버텍스 쉐이더 단계에서 미리 수행하도록 제어하는 것을 특징으로 한다.

이러한 구성에 의해, 본 발명의 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법은 PC 기반의 환경이 아닌 모바일 기반의 환경에서도 스마트 기기 내 모바일 GPU를 통해 초음파 신호에 대한 고속의 병렬처리를 수행함으로써, 의료 진단에 유용한 프레임율을 갖는 영상을 제공할 수 있는 효과가 있다.

이 발명을 지원한 국가연구개발사업

과제고유번호	NIPA-2014-H0401-14-1002
부처명	미래창조과학부
연구관리전문기관	정보통신산업진흥원
연구사업명	IT융합 고급인력과정 지원사업
연구과제명	현장진료를 위한 IT융합 휴대용 초음파 영상 시스템 개발
기 여 율	1/1
주관기관	서강대학교 산학협력단
연구기간	2012.06.01 ~ 2015.12.31

명 세 서

청구범위

청구항 1

스마트 기기가 초음파 신호를 입력 받아, 렌더 사이클(Render Cycle)을 통해 처리하여 초음파 영상을 생성하는 데 이용되는 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법에 있어서,

제1 렌더 사이클을 통해 빔포밍된 초음파 신호를 입력받아, 상기 초음파 신호로부터 DC 성분을 제거한 후, DC 성분이 제거된 초음파 신호로부터 in-phase 성분과 quadrature 성분을 나누어 각각 출력하는 단계;

제2 렌더 사이클을 통해 상기 in-phase 성분과 quadrature 성분을 갖는 초음파 신호에 대하여 quadrature 복조 및 데시메이션(decimation) 처리를 수행하는 단계;

제3 렌더 사이클을 통해 블랙홀로 간주하는 문턱치값을 입력받아, 상기 문턱치값과 상기 제2 렌더 사이클로부터 입력받은 초음파 신호의 크기를 상호 비교하여 상기 초음파 신호에 대한 블랙홀을 제거하는 단계;

제4 렌더 사이클을 통해 상기 블랙홀이 제거된 초음파 신호에 대한 에지 강화를 수행하는 단계; 및

제5 렌더 사이클을 통해 상기 에지 강화가 수행된 초음파 신호에 대하여 스캔 변환을 수행하는 단계;

를 포함하되,

상기 제1 내지 제5 렌더 사이클은

버텍스 쉐이더(vertex shader) 단계, 래스터라이저(Rasterizer) 단계 및 프래그먼트 쉐이더(fragment shader) 단계를 포함하는 그래픽스 파이프라인(Graphics pipeline) 구조로 이루어지며,

상기 스마트 기기에서 구동되는 모바일 GPU(Graphic Processing Unit)가 상기 프래그먼트 쉐이더 단계에서 수행되도록 할당된 연산과정의 일부를 상기 버텍스 쉐이더 단계에서 미리 수행하도록 제어하는 것을 특징으로 하는 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법.

청구항 2

제1항에 있어서,

상기 버텍스 쉐이더 단계는

상기 모바일 GPU가 복수 개의 정점(vertex)을 입력받고, 입력받은 정점을 이용하여 공간영역을 할당한 후, 할당한 공간영역에 대한 공간좌표를 연산하여 연산결과를 varying 변수 형태로 생성하는 것을 특징으로 하는 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법.

청구항 3

제2항에 있어서,

상기 래스터라이저 단계는

상기 모바일 GPU가 상기 버텍스 쉐이더 단계에서 출력된 varying 변수와 대응하는 스크린 내 좌표값을 검색하고, 검색한 좌표값을 varying 변수 형태로 생성하는 것을 특징으로 하는 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법.

청구항 4

제3항에 있어서,

상기 버텍스 쉐이더 단계는

상기 모바일 GPU가 상기 래스터라이저 단계에서 생성된 스크린 내 좌표값의 주변에 위치하는 좌표값을 연산하고, 연산결과를 varying 변수 형태로 생성하는 과정을 더 포함하는 것을 특징으로 하는 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법.

청구항 5

제4항에 있어서,

상기 프래그먼트 쉐이더 단계는

상기 모바일 GPU가 상기 래스터라이저 단계에서 생성된 스크린 내 좌표값에 대한 컬러를 연산하여 연산결과를 생성하는 것을 특징으로 하는 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법.

청구항 6

제5항에 있어서,

상기 모바일 GPU가 상기 프래그먼트 쉐이더 단계에서 생성된 스크린 내 좌표값에 대한 컬러 연산결과를 프레임 버퍼에 저장하는 단계를 더 포함하는 것을 특징으로 하는 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법.

청구항 7

제2항에 있어서,

상기 varying 변수는

상기 모바일 GPU의 사양에 따라 8개 내지 16개로 출력되는 것을 특징으로 하는 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법.

청구항 8

제1항에 있어서,

상기 모바일 GPU가 이전 렌더 사이클에서 생성된 이미지를 프레임버퍼에 저장하면, 저장된 이미지를 RTT(Render To Texture)기술을 이용하여 상기 모바일 GPU의 메모리에 해당하는 텍스처(Texture)로 전달하고, 전달한 이미지를 다음 렌더 사이클의 그래픽스 파이프라인 중 프래그먼트 쉐이더 단계로 전달하는 것을 특징으로 하는 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법.

청구항 9

제1항에 있어서,

상기 모바일 GPU가 고정 함수 파이프라인(Fixed Function Pipeline) 구조를 이용하여 초음파 신호를 병렬 처리하는 것을 특징으로 하는 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법.

청구항 10

제1항에 있어서,

OpenGL ES 3.0 환경에서 구현되는 것을 특징으로 하는 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방

법.

청구항 11

제1항 내지 제10항 중 어느 한 항에 따른 방법을 컴퓨터로 실행하기 위한 프로그램이 기록된 컴퓨터 판독가능 기록매체.

발명의 설명

기 술 분 야

[0001] 본 발명은 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법에 관한 것으로, 특히 휴대 가능한 스마트 기기의 모바일 GPU를 통해 대상체로부터 반사된 초음파 신호를 고속으로 병렬처리함으로써, 시간과 장소에 구애받지 않고, 신속하고, 효율적으로 초음파 신호를 처리하여 영상화할 수 있는 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법에 관한 것이다.

배 경 기 술

[0002] 최근 들어, 의료 분야에 IT 기술을 융합하여 구현되는 진단 의료 영상 분야의 기술이 급격히 발전하고 있는 추세이다. 그 중에서도 주로 사용되는 의료 초음파는 인체의 근육, 힘줄, 내부 장기, 장기의 크기, 구조 및 병리학적 손상 여부를 판단하기 위해 실시간으로 단층 영상을 가시화하는데 사용된다.

[0003] 이처럼 의료 초음파를 이용하여 진단 의료 영상을 구현하기 위해서는 대상물에 프로브를 대고 초음파를 발생시켜 상기 대상물로부터 반사된 초음파를 수신하여 영상을 구성한다. 즉, 초음파를 발생시키면 매우 짧은 시간 안에 음파가 매질 속을 지나가고, 음향 임피던스가 다른 두 매질 사이를 지날 때에는 반사파가 발생한다. 이에 발생된 반사파를 측정해 반사음이 되돌아 올 때까지의 시간을 통해 거리를 역산함으로써 진단 의료 영상을 생성하게 된다.

[0004] 이처럼 진단 의료 영상이 의료 분야에서 널리 사용됨에 따라, 남녀노소를 불문하고 많은 사용자에게 보급된 스마트 기기를 통해 시간과 장소에 상관없이 이동 중에도 초음파 신호를 이용한 진단 의료 영상을 생성하여 확인하고자 하는 요구가 점차 높아지고 있다. 하지만 이러한 요구에도 불구하고, 휴대성을 목적으로 사용되는 스마트 기기에서는 내부 CPU의 프로세싱 파워로 초음파 신호를 처리하는 경우, 실제 진단에 유용한 프레임율의 영상을 제공하기 어려운 문제점이 발생했다. 특히, 스마트 기기 내부에 GPU를 구비하더라도 PC를 기반으로 하는 환경에 비하여, CUDA, OpenCL 등과 같이 그래픽 처리 장치(GPU)에서 수행하는 병렬 처리 알고리즘을 스마트 기기의 환경에서는 PC 기반 환경과 같이 동일하게 처리하기 어려운 문제점이 발생했다.

선행기술문헌

특허문헌

[0005] (특허문헌 0001) KR 10-2012-0059740 (적어도 하나의 GPU를 구비하는 초음파 시스템, 삼성메디슨 주식회사) 2012.06.11.

발명의 내용

해결하려는 과제

[0006] 상기와 같은 종래 기술의 문제점을 해결하기 위해, 본 발명은 스마트기기가 모바일 GPU의 그래픽스 파이프라인 구조를 이용하여 초음파 신호를 처리할 때, 상기 그래픽스 파이프라인 구조 내 프래그먼트 쉐이더 단계에서 할당된 연산 처리 일부를 할당된 연산량이 상대적으로 적은 버텍스 쉐이더 단계에서 수행하도록 함으로써, 그래픽 처리 장치의 연산을 효율적으로 수행할 수 있는 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법을 제

공하고자 한다.

과제의 해결 수단

[0007] 위와 같은 과제를 해결하기 위한 본 발명의 한 실시 예에 따른 스마트 기기가 초음파 신호를 입력받아, 렌더 사이클(Render Cycle)을 통해 처리하여 초음파 영상을 생성하는데 이용되는 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법은 제1 렌더 사이클을 통해 빔포밍된 초음파 신호를 입력받아, 상기 초음파 신호로부터 DC 성분을 제거한 후, DC 성분이 제거된 초음파 신호로부터 in-phase 성분과 quadrature 성분을 나누어 각각 출력하는 단계; 제2 렌더 사이클을 통해 상기 in-phase 성분과 quadrature 성분을 갖는 초음파 신호에 대하여 quadrature 복조 및 데시메이션(decimation) 처리를 수행하는 단계; 제3 렌더 사이클을 통해 블랙홀로 간주하는 문턱치값을 입력받아, 상기 문턱치값과 상기 제2 렌더 사이클로부터 입력받은 초음파 신호의 크기를 상호 비교하여 상기 초음파 신호에 대한 블랙홀을 제거하는 단계; 제4 렌더 사이클을 통해 상기 블랙홀이 제거된 초음파 신호에 대한 에지 강화를 수행하는 단계; 및 제5 렌더 사이클을 통해 상기 에지 강화가 수행된 초음파 신호에 대하여 스캔 변환을 수행하는 단계;를 포함하되, 상기 제1 내지 제5 렌더 사이클은 버텍스 쉐이더(vertex shader) 단계, 래스터라이저(Rasterizer) 단계 및 프래그먼트 쉐이더(fragment shader) 단계를 포함하는 그래픽스 파이프라인(Graphics pipeline) 구조로 이루어지며, 상기 스마트 기기에서 구동되는 모바일 GPU(Graphic Processing Unit)가 상기 프래그먼트 쉐이더 단계에서 수행되도록 할당된 연산과정의 일부를 상기 버텍스 쉐이더 단계에서 미리 수행하도록 제어하는 것을 특징으로 한다.

[0008] 보다 바람직하게는 상기 모바일 GPU가 복수 개의 정점(vertex)을 입력받고, 입력받은 정점을 이용하여 공간영역을 할당한 후, 할당한 공간영역에 대한 공간좌표를 연산하여 연산결과를 varying 변수 형태로 생성하는 버텍스 쉐이더 단계를 수행할 수 있다.

[0009] 보다 바람직하게는 상기 모바일 GPU가 상기 버텍스 쉐이더 단계에서 출력된 varying 변수와 대응하는 스크린 내 좌표값을 검색하고, 검색한 좌표값을 varying 변수 형태로 생성하는 래스터라이저 단계를 포함할 수 있다.

[0010] 보다 바람직하게는 상기 모바일 GPU가 상기 래스터라이저 단계에서 생성된 스크린 내 좌표값의 주변에 위치하는 좌표값을 연산하고, 연산결과를 varying 변수 형태로 생성하는 과정을 더 포함하는 버텍스 쉐이더 단계를 포함할 수 있다.

[0011] 보다 바람직하게는 상기 모바일 GPU가 상기 래스터라이저 단계에서 생성된 스크린 내 좌표값에 대한 컬러를 연산하여 연산결과를 생성하는 프래그먼트 쉐이더 단계를 포함할 수 있다.

[0012] 특히, 상기 모바일 GPU가 상기 프래그먼트 쉐이더 단계에서 생성된 스크린 내 좌표값에 대한 컬러 연산결과를 프레임버퍼에 저장하는 단계를 더 포함할 수 있다.

[0013] 특히, 상기 varying 변수는 상기 모바일 GPU의 사양에 따라 8개 내지 16개로 출력될 수 있다.

[0014] 보다 바람직하게는 상기 모바일 GPU가 이전 렌더 사이클에서 생성된 이미지를 프레임버퍼에 저장하면, 저장된 이미지를 RTT(Render To Texture)기술을 이용하여 상기 모바일 GPU의 메모리에 해당하는 텍스처(Texture)로 전달하고, 전달한 이미지를 다음 렌더 사이클의 그래픽스 파이프라인 중 프래그먼트 쉐이더 단계로 전달할 수 있다.

[0015] 특히, 상기 모바일 GPU가 고정 함수 파이프라인(Fixed Function Pipeline) 구조를 이용하여 초음파 신호를 병렬처리할 수 있다.

[0016] 특히, OpenGL ES 3.0 환경에서 구현될 수 있다.

발명의 효과

[0017] 본 발명의 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법은 PC 기반의 환경이 아닌 모바일 기반의 환경에서도 스마트 기기 내 모바일 GPU를 통해 초음파 신호에 대한 고속의 병렬처리를 수행함으로써, 의료 진단에 유용한 프레임율을 갖는 영상을 제공할 수 있는 효과가 있다.

[0018] 또한 본 발명의 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법은 그래픽스 파이프라인 구조를 이용

하여 초음파 신호를 병렬 처리 시, 상기 그래픽스 파이프라인의 구조 중 프래그먼트 쉐이더 단계에서 할당된 연산량 일부를 버텍스 쉐이더 단계가 수행하도록 함으로써, 초음파 신호의 병렬 처리를 위해 수행되어야 하는 연산 처리를 나누어 처리하도록 하여 보다 신속하게 초음파 처리를 수행할 수 있는 효과가 있다.

도면의 간단한 설명

[0019] 도 1은 하나의 렌더 사이클 처리 과정을 나타낸 도면이다.

도 2는 렌더 사이클의 그래픽스 파이프라인 구조 내 연산처리 과정을 나타낸 도면이다.

도 3은 두 개의 렌더 사이클 처리 과정을 나타낸 도면이다.

도 4는 버텍스 쉐이더 단계와 프래그먼트 쉐이더 단계의 블록도이다.

도 5는 축 방향으로 4096 샘플이 넘는 크기의 데이터를 텍스처에 업로드하는 과정을 나타낸 도면이다.

도 6은 B-모드 이미징의 신호 경로를 나타낸 도면이다.

도 7은 B-모드 이미징 처리를 위해 본 발명을 적용한 렌더 사이클 처리과정을 나타낸 도면이다.

도 8은 그래픽스 파이프라인 내 9-탭 필터링 과정을 나타낸 도면이다.

도 9는 첫 번째 렌더 사이클과 두 번째 렌더 사이클 간의 초음파 신호처리과정의 일 예를 나타낸 도면이다.

도 10은 본 발명의 적용하여 획득한 B-모드 이미징의 동작 화면이다.

발명을 실시하기 위한 구체적인 내용

[0020] 이하, 본 발명을 바람직한 실시 예와 첨부한 도면을 참고로 하여 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자가 용이하게 실시할 수 있도록 상세히 설명한다. 그러나 본 발명은 여러 가지 상이한 형태로 구현될 수 있으며, 여기에서 설명하는 실시 예에 한정되는 것은 아니다.

[0021] 우선, 본 발명이 적용되는 렌더 사이클 구조에 대하여 도 1을 통해 간략히 살펴보도록 한다.

[0022] 본 발명의 구현되는 OpenGL ES 환경에서 하나의 이미지를 생성해 내는 단위를 렌더 사이클(render cycle)이라 한다.

[0023] 도 1은 하나의 렌더 사이클 처리 과정을 나타낸 도면이다.

[0024] 도 1에 도시된 바와 같이, 이러한 렌더 사이클은 빔포밍된 데이터에 대하여 초기화(initialization)과정, 렌더씬(render scene)과정, 그래픽스 파이프라인(graphics pipeline)과정을 수행하여 초음파 이미지를 생성한다.

[0025] 먼저, 초기화 과정에서는 그래픽스 파이프라인 구조 내 사용자가 프로그래밍할 수 있는 버텍스 쉐이더(vertex shader) 단계와 프래그먼트 쉐이더(fragment shader) 단계에 대하여 컴파일을 수행하고, 입력될 데이터를 저장할 그래픽 처리 장치(Graphic Processing Unit) 상의 메모리인 텍스처(texture)의 속성과 크기를 설정하며, 렌더 사이클의 출력 결과 이미지를 저장하는 프레임버퍼(framebuffer)의 속성과 크기를 설정한다.

[0026] 이어지는 렌더 씬 과정에서는 이미지가 존재하는 공간 영역을 할당하는데 사용될 정점들(vertices)을 이어지는 그래픽스 파이프라인 구조에 입력하면서 glDrawElements와 같은 명령어를 통해 그래픽스 파이프라인의 처리과정이 시작되도록 명령한다.

[0027] 이러한 그래픽스 파이프라인 처리과정은 실질적으로 그래픽 처리 장치(GPU)를 이용하여 연산의 대부분을 수행한다.

[0028] 이하, 도 2를 참조하여 실질적인 연산과정이 이루어지는 그래픽스 파이프라인 구조에 대하여 자세히 살펴보도록 한다.

[0029] 도 2는 렌더 사이클의 그래픽스 파이프라인 구조 내 연산처리 과정을 나타낸 도면이다.

[0030] 도 2에 도시된 바와 같이, OpenGL ES 3.0 환경의 그래픽스 파이프라인은 버텍스 쉐이더(vertex shader) 단계, 래스터라이저(rasterizer) 단계, 프래그먼트 쉐이더(fragment shader) 단계와 텍스처(texture) 단계로 이루어진다.

[0031] 버텍스 쉐이더 단계에서는 렌더 씬 과정을 통해 입력 받은 정점들(vertices)의 공간좌표를 계산한다. 이어서, 래스터라이저 단계에서는 스크린 상에 존재하는 픽셀 중 상기 버텍스 쉐이더 단계에서 할당한 공간 안에 존재하는 좌표를 계산해 출력한다. 이후, 프래그먼트 쉐이더 단계에서는 상기 래스터라이저 단계에서 출력된 픽셀들의 좌표를 받아 해당 픽셀들의 컬러를 계산한다. 이때, 상기 프래그먼트 쉐이더 단계에서 픽셀들의 컬러를 계산하는데 있어, GPU 메모리인 텍스처에 이전에 올려 두었던 이미지를 불러와 사용할 수 있다.

[0032] 이처럼 그래픽스 파이프라인 구조를 구성하는 단계 중 사용자가 프로그래밍이 가능한 부분은 버텍스 쉐이더 단계와 프래그먼트 쉐이더 단계이다. 이러한 상기 버텍스 쉐이더 단계와 프래그먼트 쉐이더 단계의 동작을 결정하는 쉐이더의 프로그래밍을 통해 프로그래머가 의도하는 그래픽을 GPU에서 생성해 내도록 명령할 수 있다. 상기 그래픽스 파이프라인에서 생성한 결과는 프레임버퍼(Framebuffer)에 저장이 된다.

[0033] 이러한 일련의 과정이 하나의 이미지를 생성하기 위한 하나의 렌더 사이클의 과정이다. 또한, 하나의 렌더 사이클 과정을 통해 생성된 이미지를 입력으로 받아 다른 연산을 수행하도록 하는 방법을 RTT(Render to Texture)기술이라 한다.

[0034] 이하에서는, RTT 기술을 적용한 두 개의 렌더 사이클 처리과정에 대하여 살펴보도록 한다.

[0035] 도 3은 두 개의 렌더 사이클 처리 과정을 나타낸 도면이다.

[0036] 앞서 도 2를 통해 설명한 첫 번째 렌더 사이클의 결과물이 프레임버퍼에 저장되는데 이를 다시 텍스처로 렌더링(rendering)해서 이어지는 다음 렌더 사이클에서의 입력으로 사용한다.

[0037] 이하에서는 본 발명을 이용한 초음파 신호 처리 과정의 일 예를 자세히 살펴보도록 한다.

[0038] 먼저, 그래픽스 파이프라인에서 사용할 데이터가 스마트 기기의 메인 메모리에 프레임 단위의 데이터를 그래픽 처리 장치(GPU)에서 사용되기 위해, 스마트기기의 모바일 CPU가 상기 데이터를 GPU의 메모리인 텍스처로 업로드한다. 렌더 사이클의 첫 번째 과정인 초기화 과정에서 그래픽 처리 장치(GPU)가 프레임버퍼와 텍스처 메모리를 할당하는데, 이때, 입력되는 데이터의 종류에 따라 텍스처 메모리의 속성을 다르게 정의할 수 있다. 예를 들어, 16비트 정수(integer) 형태를 갖는 빔포밍된 RF 데이터를 입력 받아 처리하는 경우에는 glTexImage2D(GL_TEXTURE_2D, 0, GL_RED, sample_no, scanline_no, 0, GL_RED_INTEGER, GL_SHORT, inputData)를 이용하여 함수로 입력될 데이터가 저장될 텍스처를 정의한다. 이때, 입력받은 데이터는 프레임 단위의 데이터이므로 2D 텍스처로 정의하기 위해 glTexImage2D 함수를 사용하고, 16 비트 정수 데이터이므로, 빨간 색에 해당되는 부분에 16비트 정수 데이터를 저장하겠다는 의미의 GL_R16I로 한다. 또한 sample_no에 프레임 데이터의 축 방향으로의 샘플 개수를 입력하고, scanline_no에 프레임 데이터의 주사선(scanline) 수를 입력하여 프레임 데이터의 크기 정보를 입력한다. 또한, GL_RED는 앞서 GL_R16I에서 빨간색만을 사용한다는 의미이고, 뒤이어 오는 GL_RED_INTEGER는 데이터 종류가 정수인 것을 나타낸다. 마지막의 inputData는 입력된 데이터를 가리키는 포인터(pointer)이다.

[0039] 이어서, 데이터의 빔포밍 수행 시, 미세 딜레이(fine delay)를 적용한 위치의 데이터를 로드하거나, 스캔 변환을 수행할 때, 2중 선형 보간법(bilinear interpolation)의 수행이 필요하다. 본 발명이 적용되는 OpenGL ES 3.0 환경에서는 텍스처의 특징을 이용해 쉐이더 프로그래밍에 2중 선형 연산을 수행하지 않고 텍스처의 정수 좌표가 아닌 곳의 데이터를 로드할 때 2중 선형 보간법을 수행한 결과를 로드할 수 있다. 렌더 사이클의 첫 번째 과정인 초기화 단계에서 프레임버퍼와 텍스처 메모리에 대한 크기 할당을 수행하는데, 여기서 텍스처의 속성을 지정해준다.

[0040] 특히, 하기와 같이 포현되는 glTexParameteri 함수를 이용하여 텍스처를 쉐이더에서 좌표 억세스를 통해 로드할 때, 2중 선형 보간법을 수행해 로드할 것인지 또는 가장 가까운 값을 로드할 것인지를 여부를 결정할 수 있다.

[0041] glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);

[0042] glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);

[0043] 이때, 상기 GL_TEXTURE_MAG_FILTER는 텍스처의 본래 사이즈보다 확대를 해서 출력하는 경우를 나타내고, GL_NEAREST는 텍스처가 본래 사이즈보다 확대된 경우에는 본래 텍스처의 크기로 채워지지 않는 부분을 가장 가까운 값으로 채우겠다는 것을 나타낸다. 이 부분을 GL_LINEAR로 표현하면 2중 선형 보간을 수행한 결과로 화면을 채우게 된다. 또한 상기 GL_TEXTURE_MIN_FILTER는 텍스처의 본래 사이즈보다 축소를 해서 화면에 출력하는 경우를 나타내고, GL_NEAREST는 이때 본래 텍스처의 값중에 화면에 값을 채울 때 접근하는 좌표에 가장 가까운 텍스처의 값으로 화면을 채우는 것을 나타내고, GL_LINEAR는 2중 선형 보간을 통해 화면을 채우는 것을 나타낸

다.

[0044] 상술한 과정을 통해 하나의 렌더 사이클 결과물은 프레임버퍼에 저장된다. 그리고 필요에 따라 프레임버퍼에 저장된 결과는 RTT(Render to Texture)기술을 이용해 이어지는 다음 렌더 사이클의 입력으로 사용될 수 있다. 렌더 사이클의 첫 번째 처리 과정인 초기화 단계에서 프레임버퍼와 텍스처 메모리 할당을 수행하는데, 그래픽 파이프라인의 출력이 저장될 프레임버퍼의 속성을 출력의 종류에 따라 속성을 다르게 정의한다. 예를 들어, Quadrature 복조를 수행해 In-phase 성분과 Quadrature 성분이 출력되는 경우에, 출력되는 In-phase 성분과 Quadrature 성분을 저장하기 위해 하기와 같이 프레임버퍼의 속성을 정의할 수 있다.

[0045] glTexImage2D(GL_TEXTURE_2D, 0, GL_RG16F, sample_no, scanline_no, 0, GL_RG, GL_FLOAT, 0);

[0046] 이때, 입력되는 데이터는 프레임 단위의 데이터이므로 2D 텍스처로 정의하기 위해, 모바일 GPU가 glTexImage2D 함수를 사용하고, 출력되는 In-phase 성분과 Quadrature 성분을 저장하는 과정에서 연산의 정확도를 위해 16bit float로 저장하며, 색상 중 빨간색과 녹색의 위치에 각각 In-phase 성분과 Quadrature 성분을 저장하기 위해 GL_RG16F로 선언한다. 또한, 모바일 GPU가 sample_no에 프레임 데이터의 축 방향으로의 샘플 개수를 입력하고, scanline_no에 프레임 데이터의 주사선 수를 입력해 프레임 데이터의 크기 정보를 입력한다. GL_RG는 앞서 GL_RG16F에서 In-phase 성분과 Quadrature 성분을 각각 픽셀의 빨간색과 녹색의 위치에 저장하기 위해 사용한다는 의미이고, 뒤이어 오는 GL_FLOAT는 데이터의 크기가 float 임을 나타낸다. glTexImage2D 함수의 마지막 위치에는 처음 입력되는 텍스처를 정의할 때는 입력되는 데이터의 포인터를 기재하나, RTT기술을 이용하기 위한 프레임버퍼를 정의할 때는 해당 프레임버퍼를 비어둔다는 의미에서 0이 입력되며, 이때 입력된 데이터를 가리키는 포인터를 기재한다.

[0047] 렌더 사이클의 두 번째 처리 과정인 렌더 씬 단계에서는 모바일 GPU가 그래픽스 파이프라인에 진입하게 하는 명령어와 함께 그래픽스 파이프라인이 저장될 프레임버퍼를 지정하는 명령을 수행한다.

[0048] OpenGL ES 3.0의 그래픽스 파이프라인 중에서 버텍스 쉐이더 단계에서 결정한 공간영역 내부에 존재하는 픽셀들의 좌표는 그래픽스 파이프라인 구조의 출력이 저장되는 프레임버퍼의 크기를 가지는 것으로 여겨 래스터라이저 단계에서 검색한다. 따라서, RTT 기술에서 렌더 타겟이 되는 텍스처인 프레임버퍼의 크기를 그래픽스 파이프라인에 입력되는 텍스처의 크기보다 작게 설정함으로써, 데시메이션 신호처리를 수행할 수 있다. 예를 들어, 축 방향으로 4096개의 샘플을 가지고 주사선 개수가 128개인 빔포밍된 데이터를 입력받아, 축 방향으로 ratio 2만큼 데시메이션을 수행하고자 하는 경우, 입력되는 빔포밍된 데이터가 저장될 텍스처를 폭이 4096, 높이가 128인 텍스처에 업로드하여 프래그먼트 쉐이더에서 로드할 수 있도록 하고, 그래픽스 파이프라인의 출력이 저장되는 프레임버퍼의 크기는 폭이 2048, 높이가 128로 속성을 설정한다. 이에 따라, 그러면 래스터라이저 단계에서는 입력된 텍스처에서 축 방향으로 한 픽셀씩 건너뛴 화면 좌표를 검색하여 프래그먼트 쉐이더 단계에 제공한다. 이러한 프래그먼트 쉐이더 단계에서는 상기 래스터라이저 단계에서 검색된 좌표를 입력 받아 입력된 텍스처에서 데이터를 축 방향으로 한 픽셀씩 건너뛴 좌표의 값을 로드한다.

[0049] 다른 예를 살펴보면, 축 방향으로 2854개의 샘플을 가지고 주사선 개수가 192개인 빔포밍된 데이터를 입력받아, 축 방향으로 ratio 2만큼 데시메이션을 수행하고자 하는 경우, 입력되는 빔포밍된 데이터가 저장될 텍스처를 폭이 2,854, 높이가 192인 텍스처에 업로드하여 프래그먼트 쉐이더 단계에서 로드할 수 있도록 하고, 그래픽스 파이프라인의 출력이 저장되는 프레임버퍼의 크기는 폭이 1,427, 높이가 192로 속성을 설정한다.

[0050] 특히, 모바일 GPU에는 두 개의 파이프라인 구조가 있는데, 첫 번째 파이프라인 구조는 고정 함수 파이프라인(fixed function pipeline)으로서, 버텍스 쉐이더 단계와 프래그먼트 쉐이더 단계에서 연산에 사용되는 물리적 프로세싱 유닛이 따로 할당되어 있다. 두 번째 파이프라인 구조는 통합 쉐이더 모델(unified shader model)로 버텍스 쉐이더 단계와 프래그먼트 쉐이더 단계에서 연산에 사용되는 프로세싱 유닛이 하나로 통합되어 있어, 사용자가 쉐이더 프로그래밍을 통해 작성한 동작에 따라 버텍스 쉐이더 단계와 프래그먼트 쉐이더 단계의 연산에 필요한 정도에 따라 프로세싱 유닛이 각각의 쉐이더 단계의 연산에 배정된다. 고정 함수 파이프라인 구조를 갖는 모바일 GPU의 종류에는 ARM사의 Mali-400 series와 Qualcomm사의 Adreno 130 등이 있다. 또한, 통합 쉐이더 모델 구조를 갖는 모바일 GPU의 종류에는 ARM사의 Mali-T600 sereis와 Qualcomm사의 Adreno 200 series, Adreno 300 series, Adreno 400 series 등이 있다. 이러한 모바일 GPU의 구조는 고정 함수 파이프라인 구조에서 통합 쉐이더 모델 구조로 바뀌어가는 추세이다.

[0051] 본 발명에 따라 버텍스 쉐이더 단계와 프래그먼트 쉐이더 단계간 로드를 분담하기 위한 최적화 방법은 사용하는 스마트 기기의 모바일 GPU가 고정 함수 파이프라인 구조인 경우에 적용하여 효과를 볼 수 있다.

[0052] OpenGL ES 3.0 환경에서의 그래픽스 파이프라인을 이용한 초음파 신호처리는 사용자가 프로그래밍 가능한 버텍스 쉐이더 단계와 프래그먼트 쉐이더 단계간의 동작을 결정하는 쉐이더 프로그래밍을 통해 구현된다. 특히, 입력되는 데이터와 출력되는 데이터의 형태 모두 직사각형 형태이므로, 버텍스 쉐이더 단계에서는 정점(vertex) 4개만을 이용해 직사각형 영역을 할당하는 것 외에는 다른 연산을 하지 않으며, 대부분의 연산은 프래그먼트 쉐이더 단계에서 수행하게 된다. 따라서, 통합 쉐이더 모델처럼 쉐이더의 프로그래밍에 따라 버텍스 쉐이더 단계와 프래그먼트 쉐이더 단계의 연산에 할당되는 프로세싱 유닛의 수가 유동적이지 않는 고정 함수 파이프라인 구조를 가지는 모바일 GPU를 사용하는 경우에는 쉐이더 프로그래밍 상에서 프래그먼트 쉐이더 단계에 할당된 연산 중 일부를 버텍스 쉐이더 단계에서 수행하도록 함으로써, 사용되지 못하고 있는 버텍스 쉐이더 단계의 연산에 할당되어 있는 프로세싱 유닛을 사용함으로써 초음파 신호의 연산 처리 속도를 높일 수 있다.

[0053] 이하, 도 4를 참조하여, 버텍스 쉐이더 단계와 프래그먼트 쉐이더의 연산과저을 자세히 살펴보도록 한다.

[0054] 도 4는 버텍스 쉐이더 단계와 프래그먼트 쉐이더 단계의 블록도이다.

[0055] 도 4에 도시된 바와 같이, 먼저 버텍스 쉐이더 단계에서 수행한 연산 결과를 varying 변수로 출력하게 되고 출력된 값은 래스터라이저 단계를 통해 스크린 상에 존재하는 값들이 검색되어 프래그먼트 쉐이더 단계로 varying 변수의 형태로 입력된다. 예를 들면, 버텍스 쉐이더 단계에서 연산한 결과는 보통 공간 영역을 나타내는 좌표가 되는데, 이때 좌표값이 varying 변수로 출력되면, 해당 렌더 사이클의 스크린 즉, 렌더 타겟인 프레임버퍼의 크기에 맞는 화면상에서 버텍스 쉐이더 단계에서 정의한 공간 영역 안에 들어가는 픽셀들의 좌표가 프래그먼트 쉐이더 단계의 varying 변수로 입력된다.

[0056] 이어서, 프래그먼트 쉐이더 단계에서는 하나의 varying 변수로 받은 좌표들로 텍스처에 업로드한 입력 데이터를 액세스해 연산에 사용한다.

[0057] 만약, 렌더 사이클의 처리 과정 내 필터링 연산이 포함되어, 프래그먼트 쉐이더 단계에서 한 픽셀의 값을 계산하기 위해서 필터의 탭 수만큼의 텍스처 데이터를 액서스할 필요가 있다면, 프래그먼트 쉐이더 단계에서는 varying 변수 형태로 입력받은 하나의 좌표에 덧셈연산하는 과정을 통해 텍스처의 이웃 좌표의 값을 계산하여 텍스처에서 이웃 좌표의 값을 로드해야 한다. 하지만 도 4에 도시된 쉐이더의 구조를 살펴보면 varying 변수는 GPU의 사양에 따라 최소 8개에서 최대 16개까지 사용될 수 있다. 즉, 모바일 GPU가 버텍스 쉐이더 단계에서 이웃 좌표에 해당될 값을 미리 연산을 하고, 그 연산 결과를 최대 16개의 varying 변수로 프래그먼트 쉐이더 단계로 전달하면 프래그먼트 쉐이더 단계에서는 이웃 좌표를 연산하기 위한 과정을 생략할 수 있다. 다만, 필터의 탭 수가 16개를 넘는 경우, 프래그먼트 쉐이더 단계에서의 이웃 좌표의 연산은 여전히 필요할 수 있다. 하지만 그렇더라도 이전보다 적은 수의 좌표연산만이 프래그먼트 쉐이더 단계에서 수행하게 된다.

[0058] 또한, 앞서 설명한 OpenGL ES 3.0 Texture의 특징을 이용한 2중 선형 보간법에서도 초음파 신호처리 과정에서 2중 선형 보간이 필요한 경우에 OpenGl ES 3.0 texture의 특징을 이용해 별다른 2중 선형 보간을 수행하기 위한 프로그래밍 없이 바로 2중 선형 보간이 가능한 것을 알 수 있다. 하지만 텍스처의 속성을 GL_LINEAR로 설정하여 2중 선형 보간을 수행하도록 한 경우와, GL_NEAREST로 설정하여 가까운 값을 가져오도록 한 경우에 연산속도가 차이가 난다. 따라서, 2중 선형 보간이 필요 없이 텍스처의 정수 좌표에서만 데이터를 로드하는 경우에는 텍스처의 속성을 GL_NEAREST로 설정하여 연산 속도를 높일 수 있다.

[0059] 이하에서는 축 방향으로 4096 sample이 넘는 크기를 갖는 데이터를 텍스처에 업로드하는 과정에 대하여 살펴보도록 한다.

[0060] 현재 상용화된 모바일 GPU는 텍스처의 크기를 폭, 높이를 각각 4096 픽셀로 제한하고 있다. 따라서 축 방향으로 4096 샘플이 넘는 크기를 갖는 초음파 신호 데이터는 지금까지 상술한 방법으로는 텍스처에 업로드하여 쉐이더 단계에서 로드해 사용할 수 없다.

[0061] 이처럼, 높이의 크기가 4096로 제한되는 것은 픽셀 기준이기 때문에, 초기화 과정에서 텍스처를 GL_RGBA로 정의하면 한 픽셀은 빨간색, 녹색, 파란색, 알파의 자리에 각각 초음파 데이터를 하나씩 넣을 수 있어, 최대 4개의 정보 저장이 가능하다. 즉, 최대 축 방향으로 16,384(4096×4) 샘플의 데이터가 입력될 수 있다.

[0062] 도 5는 축 방향으로 4096 샘플이 넘는 크기의 데이터를 텍스처에 업로드하는 과정을 나타낸 도면이다.

[0063] 도 5에 도시된 바와 같이, 축 방향으로 7808 샘플, 측면 방향으로 128 샘플을 갖는 초음파 데이터를 텍스처로 업로드한다.

[0064] 예를 들어, 주사선 0(sc0)의 축 방향으로의 데이터를 상단부터 순서대로 s0, s1, ... 라 하면, 텍스처에 저장되

는 형태는 텍스처의 속성을 GL_RG16I로 지정해 빨간색의 자리에 16bit integer인 s0, s2, s4, ... s7806, s7808을 저장하고, 녹색의 자리에 s1, s3, s5, ... s7807이 저장되도록 한다. 나머지 주사선에 대해서도 동일하게 초음파 신호 데이터가 저장되도록 한다. 프래그먼트 쉐이더 단계에서는 이와 같은 형태로 저장된 초음파 신호 데이터를 사용하기 위해서 래스터라이저 단계에서 전달되는 좌표의 픽셀 데이터를 로드한 후 빨간색, 녹색 자리 중 필요한 위치의 데이터를 골라 사용하게 된다.

[0065] 이하에서는 본 발명에 따른 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법을 이용하여 B-모드 이미징을 처리하는 과정에 대하여 자세히 살펴보도록 한다.

[0066] 먼저, 본 실시 예를 구현하는데 사용되는 대상 기기는 Google new Nexus7 (2013)이고, OS는 Android 4.4.2 KitKat이며, Qualcomm사의 Snapdragon S4 pro AP를 탑재하고 있으며, 모바일 GPU는 Adreno 320이 탑재되어 있다. 또한 본 실시 예의 개발환경은 Eclipse 4.2.1을 사용하며 Android NDK r9c 버전을 이용해 Android 어플리케이션 개발에 JAVA와 native language(C++)의 연동이 가능하도록 한다.

[0067] 도 6은 B-모드 이미징의 신호 경로를 나타낸 도면이다.

[0068] 도 6에서 사용되는 입력 데이터는 빔포밍된 데이터로 축 방향으로 2854 샘플, 주사선은 192개이며, 중심 주파수는 3.3 MHz, 샘플링율은 20 MHz로 얻은 데이터이다. 신호 경로는 일반적인 B-모드 이미징의 처리 과정을 나타낸다. 상기와 같은 특징을 갖는 입력 데이터로부터 DC 성분을 제거한 후, 디지털 시간 이득 보상(TGC: Time Gain Compensator)를 수행하고, 이어서 Quadrature 복조와 ratio 2로 데시메이션을 수행하며, 포락선 검파와 로그 압축을 수행한다. 이후, 영상의 전체적 이득(Gain)을 조절할 수 있는 이득 제어, 블랙홀 필터링, 에지 강화 및 디지털 스캔 변환(DSC: Digital Scan Conversion)을 연속하여 수행한다.

[0069] 이러한 B-모드 이미징의 신호 경로를 OpenGL ES 3.0으로 구현하기 위한 렌더 사이클은 도 7과 같이 나타나 있다.

[0070] 도 7은 B-모드 이미징 처리를 위해 본 발명을 적용한 렌더 사이클 처리과정을 나타낸 도면이다.

[0071] 먼저, 신호 경로는 하나의 신호 블록의 수행 결과가 다음 신호 블록에서 처리되기 위해서 이미 완료되어 있어야 하는 경우에 다른 렌더 사이클로 나누어진다. 예를 들어, 앞선 신호 블록을 수행 후 다음 신호 블록에서 여러 탭의 필터링을 수행하는 경우에는 그래픽스 파이프라인에서 현재 연산하고자 하는 픽셀의 값을 구하기 위해서는 현재 픽셀의 좌표에 해당하는 텍스처값 이외에 주변의 값들이 함께 필요하게 된다. 따라서, 주변의 값도 함께 액세스하기 위해서는 이미 주변의 값들이 계산된 결과가 존재해야 하므로, 렌더 사이클을 분리하여 앞선 신호 블록의 수행 결과를 프레임 단위로 프레임버퍼에 저장한 후, RTT기술로 텍스처화 하여 다음 렌더 사이클에서 입력으로 사용한다.

[0072] 또한 하나의 렌더 사이클 내부의 연산과정은 동시에 수행되므로, 렌더 사이클의 개수를 최소화로 설정하는 것이 성능 향상에 유리하다. 이때, 랜더 사이클에서 수행되는 연산은 출력 결과물이 저장될 버퍼 상의 픽셀 수만큼 수행될 수 있다.

[0073] 상술한 기준을 통해 나누어지는 각각의 렌더 사이클 중에서, 제1 렌더 사이클을 자세히 살펴보도록 한다.

[0074] 제1 렌더 사이클에서는 빔포밍된 데이터로부터 DC 성분을 제거하고, Quadrature 복조과정의 일부를 수행하여 데이터를 출력(x_i, x_q)한다. 이때, 상기 DC 성분을 제거하는 DC 성분 제거 신호 블록은 32-탭 힐버트 필터(Hilbert Filter)를 포함하고, 이러한 필터를 통해 고역 통과 필터링을 수행하여 빔포밍된 데이터로부터 DC 성분을 제거할 수 있다.

[0075] 도 8은 그래픽스 파이프라인 내 9-탭 필터링 과정을 나타낸 도면이다.

[0076] 도 8에 도시된 바와 같이, 프래그먼트 쉐이더 단계에서는 프레임버퍼에 존재하는 픽셀의 수만큼 프래그먼트 쉐이더 단계의 main() 함수를 반복적으로 수행한다. 상기 main()함수가 한 번 수행할 때마다 프레임버퍼의 해당 픽셀 좌표의 컬러값이 결정된다. 따라서 필터링을 수행하는 경우에는 하나의 main()안에서 현재 연산중인 프레임버퍼의 픽셀 좌표에 해당하는 텍스처(입력 이미지) 상의 이웃 좌표의 데이터가 필요하게 된다. 따라서, 도 8을 자세히 살펴보면 현재 연산중인 픽셀의 좌표가 (x, y)라면 p0-p8의 값을 y축의 값의 계산을 통해 텍스처 상의 이웃 데이터를 texture()함수로 로드한다. 이후, 필터 계수와 로드한 p0-p8의 컨벌루션(convolusion) 연산이 수행되고, 그 결과가 Pout으로 출력되어 프레임버퍼상에서 해당 좌표에 저장된다. 또한 다음 픽셀의 값을 계산하기 위해서는 프레임버퍼의 다음 좌표를 기준으로 main()을 반복하여 수행한다. 이와 같은 동일한 방법으로 main()을 프레임버퍼에 존재하는 픽셀 수만큼 반복적으로 수행해 필터링을 수행한 결과를 프레임 단위로 프레임

버퍼에 저장한다. 같은 방법으로 B-모드 이미징의 예시에서는 16-탭 고역 통과 필터링을 수행한다.

[0077] 이후, Quadrature 복조 신호 블록에서 cos과 sin을 곱해 in-phase 성분과 quadrature 성분을 출력하는 과정까지 제1 렌더 사이클에서 신호처리를 수행한다. 상기 제1 렌더 사이클의 출력은 in-phase 성분과 quadrature 성분 두 가지이기 때문에 제1 렌더 사이클의 출력이 저장될 프레임버퍼는 GL_RG16F로 하여 빨간색 자리에 in-phase 성분 저장하고, 녹색 자리에 quadrature 성분을 저장하도록 한다. 프레임버퍼의 크기는 입력과 동일하다. 상기 제1 렌더 사이클의 출력 데이터를 제2 렌더 사이클로 전달할 때는 연산의 정확도를 유지하기 위해 16bit float 형태로 프레임버퍼에 저장한다.

[0078] 즉, 상기 제1 렌더 사이클에서는 2854 × 192 픽셀 크기를 갖는 빔포밍된 데이터가 입력되어, 2854 × 192 = 547,968 회의 연산을 거쳐 2854 × 192 픽셀의 출력 데이터 x_i, x_q를 출력한다.

[0079] 이어서, 제2 렌더 사이클에서는 상기 제1 렌더 사이클에서 프레임버퍼에 저장한 in-phase 성분과 quadrature 성분을 RTT기술을 이용해 입력 텍스처로 받아 데시메이션 신호처리를 수행한다. 데시메이션율이 2 이므로, 제2 렌더 사이클의 결과가 저장될 프레임버퍼의 폭을 2854의 절반인 1427로 설정한다. 또한, 저역 통과 필터링이 Quadrature 복조 과정과 데시메이션 과정에 중복적으로 필요하므로 각 과정에서의 저역 통과 필터링을 한 번에 수행할 있도록 필터를 설계한다. 따라서 입력받은 데이터에 대하여 저역 통과 필터링을 수행하고, 포락선 검파를 square-root를 이용해 수행한 후 로그 압축을 수행하고, 최종적으로 이득을 곱해 전체적인 영상의 이득을 조절할 수 있도록 한다. 상기 제2 렌더 사이클의 결과는 포락선 검파 과정을 거치면서 성분이 하나로 합성되므로, 출력 데이터가 저장될 프레임버퍼는 GL_R16F로 빨간색 자리에 16 bit float로 저장된다.

[0080] 즉, 제2 렌더 사이클에서는 앞서 제1 렌더 사이클에서 출력된 2854×192 픽셀 크기를 갖는 데이터 (x_i, x_q)를 입력받아, 나머지 quadrature 복조 과정을 수행한다. Quadrature 복조과정 뒤와 데시메이션 과정 앞에 들어가는 안티-앨리어싱(anti-aliasing) 필터(LPF)는 quadrature 복조와 데시메이션 사이에서 한번 수행한다. 이어서, 입력된 데이터에 대하여 square-root를 이용해 포락선 성분을 검출하고, 로그 압축을 통해 스케일을 맞춰준다. 마지막으로 전체적인 게인을 곱해 전체 이득을 제어하여 1427×192 픽셀 크기를 갖는 로그 압축 데이터(Log compressed data)를 출력한다. 이때, 상기 로그 압축 데이터를 출력하기 위해, 총 1427 X 192 = 273,984 회의 연산이 수행된다.

[0081] 제3 렌더 사이클에서는 3×3 평균화 필터를 이용해 블랙홀 필터링을 수행한다. 평균화를 수행하는 정도를 설정하기 위해, 사용자로부터 해당 픽셀의 값을 블랙홀로 간주하는 정도를 나타내는 문턱치값을 입력받아 문턱치값 3×3의 윈도우의 중앙에 위치한 값이 상기 문턱치 값 이하일 때만 평균화한 결과를 출력하고, 그렇지 않은 경우에는 기존의 중앙에 위치한 값을 출력하도록 한다. 이때, 상기 3×3 평균화 필터의 경우에도 한 픽셀의 값을 연산하는데 있어, 이웃 좌표의 값이 필요하므로 앞선 렌더 사이클의 연산이 모두 끝난 뒤에 수행되어야 함에 따라 렌더 사이클을 분리한다.

[0082] 즉, 제3 렌더 사이클은 앞서 제2 렌더 사이클에서 출력된 1427×192 픽셀 크기를 갖는 로그 압축 데이터(Log compressed data)를 입력받아, 입력 받은 로그 압축 데이터에 대하여 3×3 평균화 필터링을 통해 블랙홀을 제거하고, 이에 따라, 1427 X 192 픽셀 크기를 갖는 블랙홀이 필터링된 데이터(Blackhole filtered data)를 출력한다. 이때, 상기 블랙홀이 필터링된 데이터를 출력하기 위해 수행되는 연산횟수는 총 1427 X 192 = 273,984 회이다.

[0083] 이후, 제4 렌더 사이클에서는 3×3 소벨 필터(Sobel filter)를 이용해 데이터의 윤곽선을 검출하여 에지 강화를 수행한다. 상기 소벨 필터링을 통해 에지 성분을 검출한 영상과 기존 영상을 더하여 에지 강화가 이루어지는데, 검출한 에지 성분의 영상을 더하는 정도를 나타내는 가중치 변수를 사용자로부터 입력받아 에지 강화의 강도를 설정한다. 특히, 3×3 소벨 필터의 경우도 한 픽셀의 값을 연산하는데 이웃 좌표의 값이 필요하므로, 앞선 렌더 사이클의 연산이 모두 끝난 뒤에 수행되어야 함에 따라, 렌더 사이클을 분리한다.

[0084] 즉, 제4 렌더 사이클은 상기 제3 렌더 사이클로부터 출력된 1427 X 192 픽셀 크기를 갖는 블랙홀이 제거된 데이터(blackhole filtered data)를 입력받아, 3x3 소벨 필터링(Sobel filtering)을 통해 에지 성분을 검출하여 에지 강화를 수행함으로써, 1427×192 픽셀 크기를 갖는 에지가 강화된 데이터(Edge enhanced data)를 출력한다. 이때, 상기 에지가 강화된 데이터가 출력되기 위해 연산되는 횟수는 총 1427×192 = 273,984 회이다.

[0085] 이어서, 제5 렌더 사이클에서는 스캔 변환을 수행한다. 본 실시 예에서 사용하는 데이터는 컨벡스 프로브(convex probe)로부터 획득한 데이터이므로 컨벡스 스캔 변환을 수행한다. 이때, 상기 스캔 변환이 필터링의 경우처럼 이웃좌표의 값이 필요하지 않음에도 렌더 사이클을 분리한 것은 하나의 렌더 사이클에서의 연산 개수는

렌더 타겟의 픽셀 수에 비례하기 때문이다. 따라서, 스캔 변환의 출력은 폭이 830 픽셀, 높이가 640 픽셀이므로 한 픽셀의 연산에 필요한 연산량을 α라 하면 총 830×640×α(531200α) 만큼의 연산이 필요하다.

[0086] 즉, 제5 렌더 사이클은 앞서 제4 렌더 사이클로부터 출력된 1427×192 픽셀 크기를 갖는 에지가 강화된 데이터(Edge enhanced data)를 입력받아, 입력 받은 데이터에 대하여 디지털 스캔 변환을 통해 830×640 픽셀 크기를 갖는 최종 출력 이미지 (Digital scan conversed image)를 생성하여 출력한다. 이때, 연산수는 830 X 640 = 531,200 회가 된다.

[0087] 반면 제4 렌더 사이클의 경우에는 1427×192×α(273984α) 만큼의 연산을 필요로 하게 되는데, 만약 렌더 사이클을 분리하지 않고, 한 렌더 사이클에서 같이 연산을 수행하는 경우, 한 렌더 사이클에서는 연산이 동시에 이루어지게 되므로, 필요 이상으로 소벨 필터링을 수행하게 되는 문제가 발생하므로, 이를 방지할 수 있다.

[0088] 따라서, 필요하지 않은 연산수를 줄이기 위해 출력 프레임 버퍼의 픽셀 수를 고려하여 렌더 사이클을 분리하는 것이 중요하다.

[0089] 도 10은 본 발명의 적용하여 획득한 Nexus7에서의 B-모드 이미징의 동작 화면이다.

[0090] 도 10에 도시된 바와 같이, 시간 이득 보상(TGC)을 8단계로 나누어 각 깊이별로 영상의 밝기를 사용자가 조절가능하게 하였으며, 전체적인 이득(overall gain) 또한 조절 가능하도록 하였다. 평균화 필터링의 정도를 설정하는 문턱치의 값과 에지 강화의 정도를 설정하는 가중치 값을 사용자가 조절할 수 있도록 사용자 인터페이스를 만든다.

[0091] 또한, 이러한 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법은 컴퓨터로 실행하기 위한 프로그램이 기록된 컴퓨터 판독가능 기록매체에 저장될 수 있다. 이때, 컴퓨터가 읽을 수 있는 기록매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록 장치를 포함한다. 컴퓨터가 읽을 수 있는 기록 장치의 예로는 ROM, RAM, CD-ROM, DVD±ROM, DVD-RAM, 자기 테이프, 플로피 디스크, 하드 디스크(hard disk), 광 데이터 저장장치 등이 있다. 또한, 컴퓨터가 읽을 수 있는 기록매체는 네트워크로 연결된 컴퓨터 장치에 분산되어 분산방식으로 컴퓨터가 읽을 수 있는 코드가 저장되고 실행될 수 있다.

[0092] 상술한 바와 같이, 본 발명을 통해 PC 환경에서처럼 스마트 기기에서도 초음파 신호처리를 GPU를 이용해 고속의 병렬형태로 수행할 수 있다. 이에 더하여 초음파 영상 시스템의 초소형화에 기여할 수 있다. 또한 남녀노소를 불문하고 널리 보급화된 스마트 기기에 단순히 어플리케이션 형태의 소프트웨어를 설치하고, 이를 통해 초음파 영상을 획득할 수 있다는 점에서 초음파 시스템의 비용 절감과 대중화에 기여할 수 있다. 초음파 영상에 대한 정보를 담고 있는 데이터를 원격으로 전달받아 스마트 기기에서의 신호처리를 통해 초음파 영상을 얻을 수 있다는 점에서 원격 진료의 보급화 또한 기대할 수 있다.

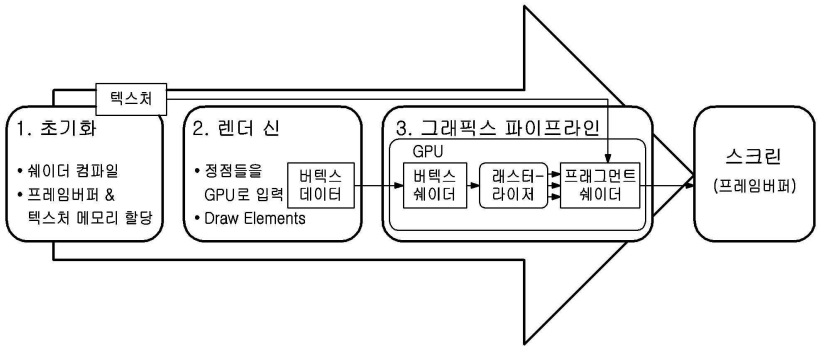
[0093] 본 발명의 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법은 PC 기반의 환경이 아닌 모바일 기반의 환경에서도 스마트 기기 내 모바일 GPU를 통해 초음파 신호에 대한 고속의 병렬처리를 수행함으로써, 의료 진단에 유용한 프레임율을 갖는 영상을 제공할 수 있는 효과가 있다.

[0094] 또한 본 발명의 스마트 기기를 이용한 초음파 신호의 고속 병렬 처리 방법은 그래픽스 파이프라인 구조를 이용하여 초음파 신호를 병렬 처리 시, 상기 그래픽스 파이프라인의 구조 중 프래그먼트 쉐이더 단계에서 할당된 연산량 일부를 버텍스 쉐이더 단계가 수행하도록 함으로써, 초음파 신호의 병렬 처리를 위해 수행되어야 하는 연산 처리를 나누어 처리하도록 하여 보다 신속하게 초음파 처리를 수행할 수 있는 효과가 있다.

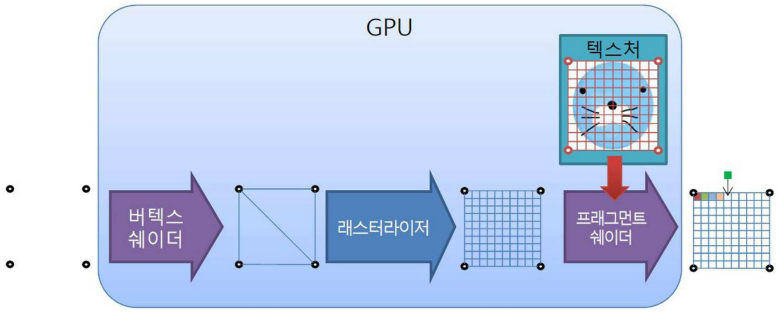
[0095] 상기에서는 본 발명의 바람직한 실시 예에 대하여 설명하였지만, 본 발명은 이에 한정되는 것이 아니고 본 발명의 기술 사상 범위 내에서 여러 가지로 변형하여 실시하는 것이 가능하고 이 또한 첨부된 특허청구범위에 속하는 것은 당연하다.

도면

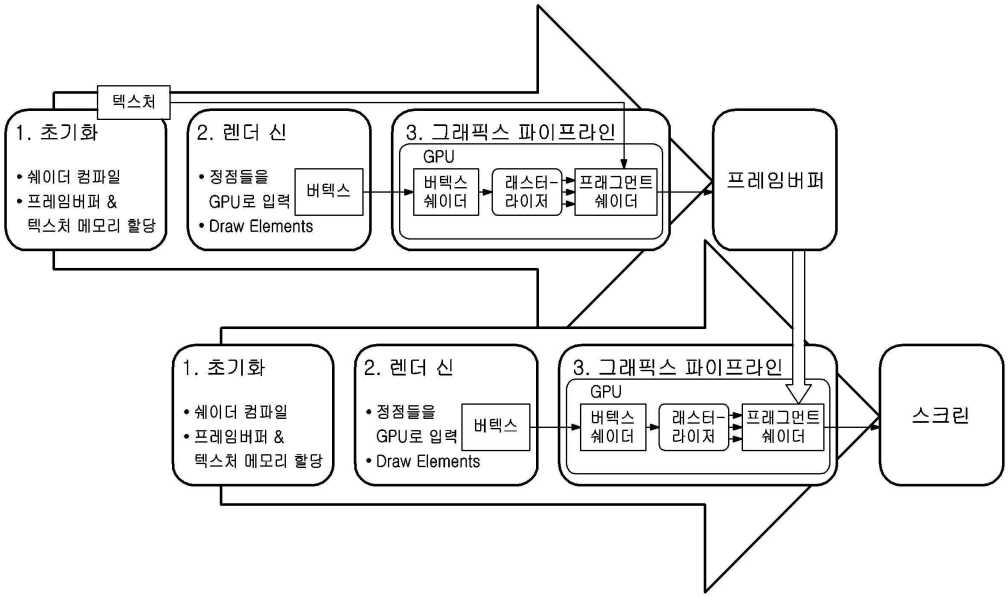
도면1



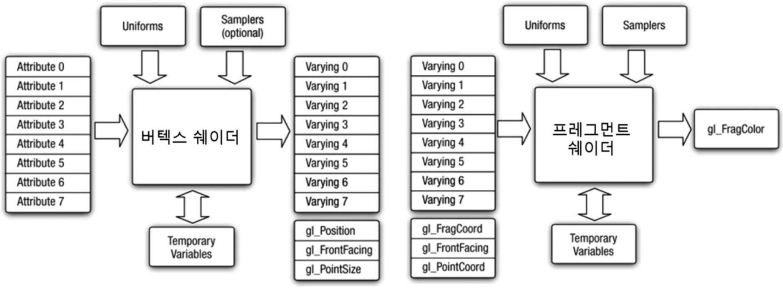
도면2



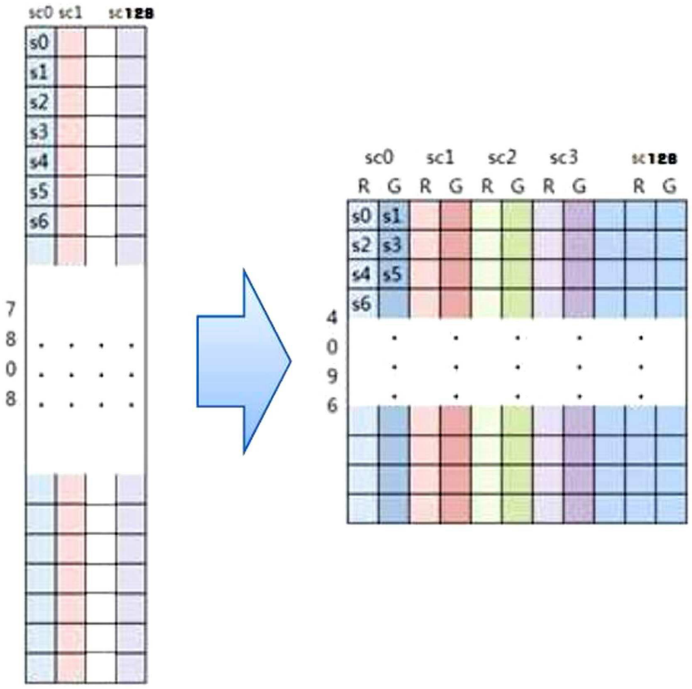
도면3



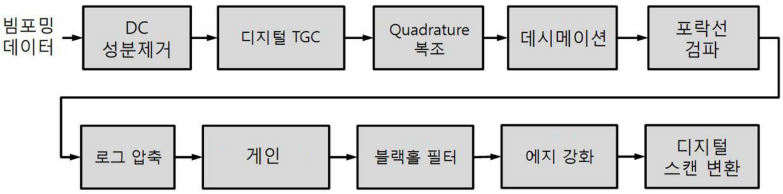
도면4


Uniforms
Samplers (optional)
Attribute 0
Attribute 1
Attribute 2
Attribute 3
Attribute 4
Attribute 5
Attribute 6
Attribute 7
버텍스 쉐이더
Temporary Variables
Varying 0
Varying 1
Varying 2
Varying 3
Varying 4
Varying 5
Varying 6
Varying 7
gl_Position
gl_FrontFacing
gl_PointSize
Uniforms
Samplers
Varying 0
Varying 1
Varying 2
Varying 3
Varying 4
Varying 5
Varying 6
Varying 7
gl_FragCoord
gl_FrontFacing
gl_PointCoord
프레그먼트 쉐이더
Temporary Variables
gl_FragColor

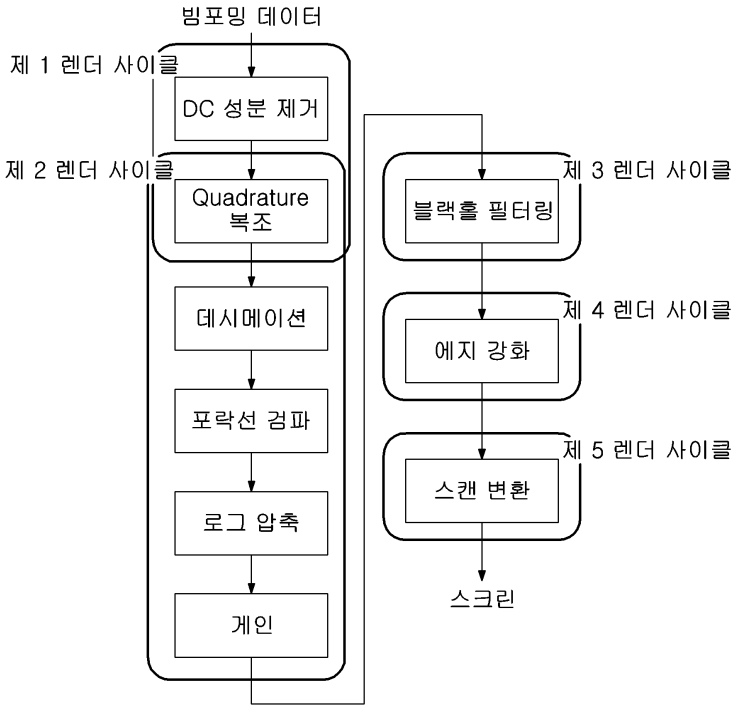
도면5


sc0 sc1 sc128
s0
s1
s2
s3
s4
s5
s6
7
8
0
8
sc0 sc1 sc2 sc3 sc128
R G R G R G R G R G
s0 s1
s2 s3
s4 s5
s6
4
0
9
6

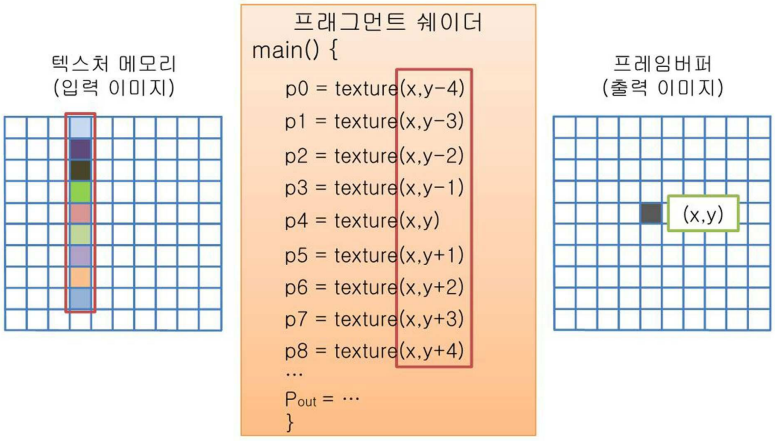
도면6


빔포밍 데이터
DC 성분제거
디지털 TGC
Quadrature 복조
데시메이션
포락선 검파
로그 압축
게인
블랙홀 필터
에지 강화
디지털 스캔 변환

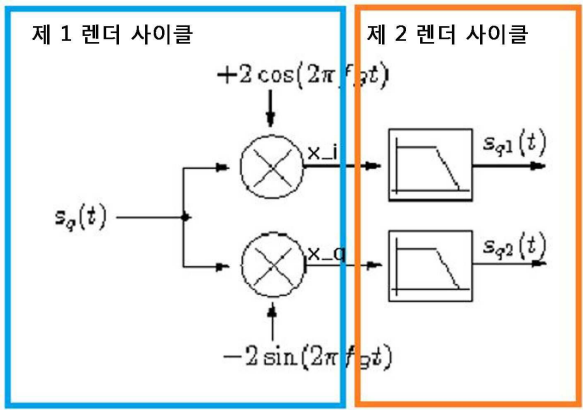
도면7


빔포밍 데이터
제 1 렌더 사이클
DC 성분 제거
제 2 렌더 사이클
Quadrature 복조
데시메이션
포락선 검파
로그 압축
게인
제 3 렌더 사이클
블랙홀 필터링
제 4 렌더 사이클
에지 강화
제 5 렌더 사이클
스캔 변환
스크린

도면8


텍스처 메모리
(입력 이미지)
프래그먼트 쉐이더
main() {
p0 = texture(x,y-4)
p1 = texture(x,y-3)
p2 = texture(x,y-2)
p3 = texture(x,y-1)
p4 = texture(x,y)
p5 = texture(x,y+1)
p6 = texture(x,y+2)
p7 = texture(x,y+3)
p8 = texture(x,y+4)
…
Pout = …
}
프레임버퍼
(출력 이미지)
(x,y)

도면9



도면10

专利名称(译)	一种使用智能设备的超声波信号的高速并行处理方法		
公开(公告)号	KR1020160026110A	公开(公告)日	2016-03-09
申请号	KR1020140114032	申请日	2014-08-29
[标]申请(专利权)人(译)	서강대학교산학협력단		
申请(专利权)人(译)	서강대학교산학협력단		
[标]发明人	SONG TAI KYONG 송태경 KONG UKYU 공우규		
发明人	송태경 공우규		
IPC分类号	A61B8/14 A61B8/08 G06F17/30 G01S15/89 G06T15/08 A61B8/00 G01S7/52		
CPC分类号	A61B8/5223 G06F17/30519 G01S15/8977 G06T15/08 A61B8/4427 G01S7/52047 A61B8/5207 A61B8/463 A61B8/14 A61B8/00 G16H50/50 H04L27/1525 G06F16/24569 G06F9/28		
其他公开文献	KR101771242B1		
外部链接	Espacenet		

摘要(译)

本发明是一种智能设备上使用智能设备的超声信号的高速并行处理方法被用于接收超声波信号，并通过一个渲染循环处理（渲染周期）产生的超声波图像，在第一渲染周期从超声信号中去除DC分量并从去除DC分量的超声信号输出同相分量和正交分量;通过第二渲染周期对具有同相分量和正交分量的超声信号执行正交解调和抽取处理;通过输入周期中接收所述第三渲染认为是孔的阈值，其包括相互比较超声波信号的幅度从阈值接收和所述第二渲染周期除去黑洞用于所述超声波信号的步骤;对通过第四渲染周期去除黑洞的超声信号执行边缘增强;并且通过第五渲染周期对边缘增强超声信号执行扫描转换，其中第一至第五渲染周期包括顶点着色器步骤，光栅化器，以及包括片段着色器步骤的图形管线结构，其中由智能设备驱动的移动GPU（图形处理单元）被分配以在片段着色器步骤中执行控制顶点着色器的一部分以在顶点着色器步骤中预先执行。根据该结构，通过我的移动GPU智能设备使用本发明的智能装置的超声波信号的在基于移动环境比通过进行超声信号的高速并行处理的基于PC的环境中的其他高速并行处理方法中，可以提供具有对医学诊断有用的帧速率的图像。 支持本发明的国家研发项目 作业号码 NIPA-2014-H0401-14-1002 bucheomyeong 未来创造科学系 研究管理专业 信息通信产业振兴机构 研究项目名称 IT融合先进的人力资源课程支持业务 研究项目名称 开发用于现场医疗的IT融合便携式超声成像系统 1.1 主要组织 西江大学工业学院合作基金会 研究期 2012年6月1日 - 2015年12月31日