[UCI]I804:24011-200000236515

2008년 8월

석사학위 논문

고성능 *3D* 지오메트리 연산을 위한 *CORDIC* 기반 오프라인 벡터회전 알고리즘 연구

조선대학교 대학원

컴 퓨 터 공 학 과

김 은 옥

고성능 *3D* 지오메트리 연산을 위한 *CORDIC* 기반 오프라인 벡터회전 알고리즘 연구

CORDIC-Based Off-line Vector Rotation Algorithm for High Performance 3D Geometry Operations

2008년 8월 25일

조선대학교 대학원

컴 퓨 터 공 학 과

김 은 옥

고성능 *3D* 지오메트리 연산을 위한 *CORDIC* 기반 오프라인 벡터회전 알고리즘 연구

지도교수 이 정 아

이 논문을 공학석사학위신청 논문으로 제출함.

2008년 4월

조선대학교 대학원

컴 퓨 터 공 학 과

김 은 옥

김은옥의 석사학위논문을 인준함

위원장　조선대학교 교수 _____(인)

위　원　조선대학교 교수 _____(인)

위　원　조선대학교 교수 _____(인)

2008년　5월

조선대학교 대학원

목 차

그림목차

표목차

ABSTRACT

CORDIC-Based Off-line Vector Rotation Algorithm for High Performance 3D Geometry Operations

Kim, Eun Ok
Advisor : Prof. Lee, Jeong-A
Department of Computer Engineering
Graduate School of Chosun University

Recently, as 3D computer graphics applications are rapidly growing, many works for efficient 3D graphic processor are suggested. Those works are mostly focused on high performance. But, on applications, such as mobile and ubiquitous, we need to consider power and cost problems.

COordinate Rotation DIgital Computer (CORDIC) is a well-known algorithm for calculating transcendental functions in a fast and efficient manner. Traditional hardware for calculating transcendental arithmetic operations are complex in nature, large in size and high power requirement. In contrast, CORDIC algorithm can calculate such complex arithmetic operations with hardware that require just adders and shifters. This has made CORDIC the algorithm of choice for Digital Signal Processing(DSP) computing engine.

In a similar vein, we can expect better performance by applying CORDIC-type primitives to the main operations of 3D computer graphics. These CORDIC-type primitives are especially useful in Rigid Body Rotations, Interpolation of Orientations, etc.

2D CORDIC algorithm has 'N' elementary angles determined by required

accuracy and performs rotations with that number of angles. With a high number of iterations we have a high accuracy and low performance while with a small number of iterations we have low accuracy and high performance. This problems exist in CORDIC-3D that was extended from 2D CORDIC.

In this paper, we propose two off-line searching methods named ABS-3D and SCS-3D based on MVR-CORDIC, MSR-CORDIC algorithms to improve operating performance of CORDIC-3D algorithm.

ABS-3D and SCS-3D are off-line methods, so we can use them for such applications like graphic movies that are not required dynamic inputs. From this methods, we can get the best sequence for the given inputs and reduce the number of iterations. Using ABS-3D and SCS-3D, we reduced 66.7%, 50% in the amount of operations, respectively.

Ⅰ. 서 론

3D 컴퓨터 그래픽 연산을 위한 고성능 그래픽 프로세서의 필요성이 증가하면서 이를 만족시키기 위한 다양한 연구가 이루어지고 있다. 과거에는 3D 그래픽스 처리의 전 과정을 CPU 프로그래밍을 기반으로 가속시켜 왔기 때문에 CPU의 부담이 가중되었다. 그러나 입체감과 사실감을 높이기 위해 더욱 향상된 그래픽 처리 능력이 요구되면서, 이를 위한 하드웨어 가속 구조가 개발되었다.

90년대 중반에 등장한 초기 PC용 GPU는 래스터 처리 단계만을 가속할 수 있었지만, 이후 지오메트리 연산 과정까지 포함된 GPU가 개발되면서 3D 그래픽의 처리 성능은 급속도로 발전하여 왔다. 특히, 이러한 기술들이 최근에는 점차 휴대용 모바일 기기에도 적용되어 가고 있는 추세이다. PC용 GPU는 하드웨어를 병렬로 배치하여 성능을 높일 수 있었으나, 휴대용 단말에서는 하드웨어 면적과 전력공급이 제한적인 특성 때문에 저전력 및 저비용의 3D 그래픽 처리를 위한 단순하고 효율적인 구조가 필요하게 되었다.

이러한 조건을 만족하기 위해 Lang등은 특유의 하드웨어 간결성을 갖는 CORDIC(COordinate Rotational DIgital Computer) 알고리즘을 3차원으로 확장한 CORDIC-3D 알고리즘을 제안하였다[1]. 이 연구의 바탕이 된 CORDIC은 삼각함수를 비롯한 여러 가지 초월 함수의 계산을 효율적으로 할 수 있는 연산기술로 알려져 있다. Jack E. Volder에 의해 고안되었으며, 쉬프트와 덧셈만을 이용하여 2차원 벡터회전과 같은 복잡한 연산을 수행할 수 있기 때문에 벡터연산을 기반으로 하는 다양한 분야에서 활용되고 있다[2].

Lang의 연구에서는 3차원상의 벡터회전을 CORDIC-type의 프리미티브(primitives)로 수행할 수 있도록 함으로써 벡터회전 연산이 큰 비중을 차지하는 3D 지오메트리 연산 과정에서 성능을 향상시켰다. 기존의 벡터회전 연산이 행렬의 곱과 같은 대수적(algebraic)인 형태로 표현되는데 반하여, CORDIC 알고리즘을 이용한 벡터회전의 표현은 기하학적 성질을 반영함으로써 유연성을 얻을 수 있고 다양한 산술연산을 고성능으로 수행할 수 있었다.

2차원 기본 CORDIC 알고리즘은 정밀도에 의해 결정되는 기본각의 개수 'N'을 전체 반복횟수로 가지며, 그만큼의 회전을 모두 수행해야 하므로 계산 속도가 늦다는 단점이 있어 성능 면에서 비효율적일 수 있었다. 또한 높은 정확도를 요구하는 시스템 일수록 데이터의 비트 폭이 커지게 되어 큰 반복횟수를 가질 수 밖에 없다. 이러한 단점은 기본 CORDIC을 3차원으로 확장시킨 CORDIC-3D 알고리즘에서도 그대로 나타난다.

본 논문에서는 2차원 CORDIC 알고리즘을 개선하기 위해 제안된 오프라인 알고리즘인 MVR(Modified Vector Rotational)-CORDIC, MSR(Mixed Scaling Rotation)-CORDIC과 같이 오프라인 상에서 사전에 주어진 목표위치가 있을 때, 최소의 반복만으로 해당 회전을 수행 할 수 있도록 최적의 시퀀스를 검색하는 방법을 통해 CORDIC-3D를 개선하고자 한다.

본 논문에서 제안하는 알고리즘은 사전에 알려진 입력에 대해서만 적용할 수 있기 때문에 3D 그래픽스 처리 중에서도 동적으로 입력되는 데이터에 대한 처리가 요구되는 응용분야가 아닌, 그래픽 영화와 같이 데이터가 미리 결정되어 영화 시청 시에 반복적으로 사용되는 경우에 적용할 수 있다. 예를 들어 일반적으로 MPEG 영상에서 데이터 압축을 위해 실제 데이터를 저장하기 보다는 데이터간의 관계인 모션 벡터를 저장하는 것과 같은 아이디어로 볼 수 있다.

그래픽에서 회전 연산을 수행할 때, 기존의 CORDIC 방식을 이용하여 많은 수의 기본각 회전을 하는 대신 회전이 적용될 원 데이터와 회전후의 데이터간의 관계를 나타내는 회전각을 개선된 CORDIC-3D 알고리즘에 의해서 미리 계산된 최소의 기본각들로 표현한다. 그리고 실제로 회전할 때에 이 계산된 최소의 기본 각들만을 이용하여 적은 반복으로 회전을 수행할 수 있도록 한다.

이와 같이 사전에 입력이 결정된 그래픽 응용 분야에서 주어진 입력에 대한 최적의 시퀀스를 결정하여 이용함으로써 기존의 CORDIC-3D 연산의 문제점인 반복횟수를 현저히 줄여 3차원 벡터회전 연산을 빠르게 할 수 있도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문과 관련된 3차원 그래픽 처리 과정과 2차원 기본 CORDIC 알고리즘, 3차원으로 확장된 CORDIC-3D 알고리즘에 대한 관련 연구 내용을 고찰하고, 3장에서는 본 논문에서 제안하는 개선된

CORDIC-3D 오프라인 알고리즘에 대한 설명과 함께 실험 결과와 설계 요소에 관한 내용을 기술한다. 4장에서는 결론과 향후 연구 과제를 제시한다.

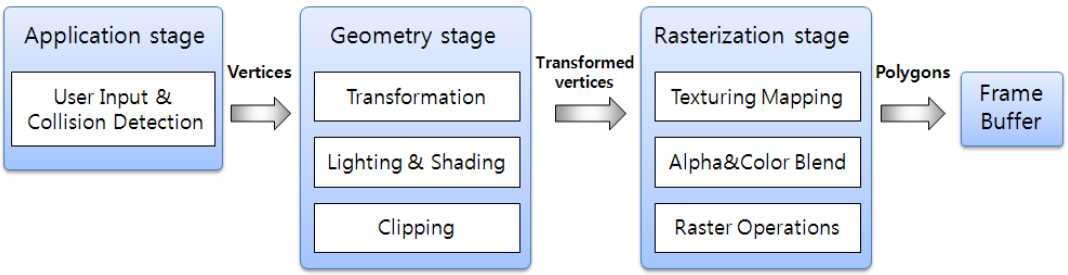
Ⅱ. 관련연구

*1. 3*차원 그래픽 처리 과정

3차원 그래픽 처리는 특성상 많은 양의 수학적 계산이 요구되기 때문에 그래픽 프로세서의 설계의 핵심은 높은 대역폭의 데이터 스트림이며, 이 데이터 스트림을 통해 그래픽 요소들을 처리할 수 있는 강력한 연산 기능을 제공하게 된다[3],[4].

3차원 그래픽 처리 과정은 (그림 1)과 같이 크게 어플리케이션 처리 단계, 지오메트리 단계, 래스터화 단계로 이루어진다[5]. 지오메트리 단계에서는 정점 데이터의 위치변환과 조명에 관련된 부분에 해당하는 기하학적 연산을 수행한다. 변환된 데이터가 전달되면 래스터화 단계에서는 프레임 버퍼에 저장될 픽셀 값을 결정하고, 텍스처 매핑, 알파 블랜드, 컬러 블랜드와 같은 작업들을 수행한다[6].

현재 지오메트리 단계내의 정점 연산을 수행하는 정점 프로세서는 부동소수점 파이프라인이 적용된 벡터곱셈, 역수와 역 제곱근 또는 지수, 삼각함수 등 다양한 연산을 하기 위한 장치들을 포함하고 있다. 이 과정에서는 데이터의 특성상 변환될 좌표를 대상으로 처리가 이루어지며 이러한 연산들은 대부분 좌표계 변환으로 행렬의 곱셈과 같은 대수적 계산으로 처리할 수 있다[9],[10]. 이처럼 다양한 기능이 요구되기 때문에 성능조건을 만족하기 위해서는 고성능의 연산을 수행하는 모듈이 지원되어야 한다[7],[8].



(그림 1) 3차원 그래픽 처리과정

2. *CORDIC* 알고리즘

CORDIC은 삼각함수를 비롯한 초월 함수 계산을 효율적으로 할 수 있는 연산 기술이다. Jack E. Volder에 의해 고안 되었으며, 복잡한 산술 연산을 덧셈과 쉬프트 연산만으로 수행할 수 있도록 하여 계산을 빠르게 한다. 구현 면에서도 쉬프터와 가산기만을 이용하여 하드웨어를 간단하게 구현할 수 있다는 장점이 있어 하드웨어 면적의 비용을 줄일 수 있기 때문에 다양한 분야에서 활용되고 있다.

2.1 CORDIC의 원리

일반적인 2차원 회전에서 벡터 $[x\ y]^T$를 각 Φ에 대해 회전한 후 벡터 $[x'\ y']^T$는 식 (1)과 같이 나타낼 수 있다[11].

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Phi & -\sin\Phi \\ \sin\Phi & \cos\Phi \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \tag{1}$$

위의 식 (1)의 회전 행렬을 변형하여 다음과 같이 표현 가능하다.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \cos\Phi \begin{bmatrix} 1 & -\tan\Phi \\ \tan\Phi & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \tag{2}$$

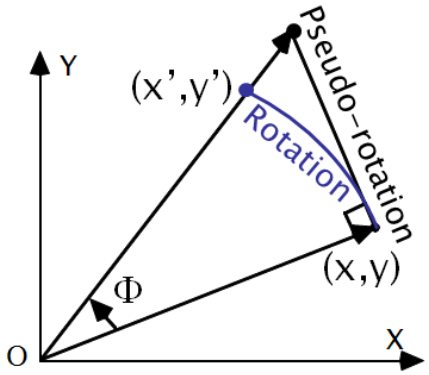
CORDIC 알고리즘은 하드웨어 구조를 단순화 하기위해 식 (2)의 $\tan\Phi$를 쉬프트 연산으로 처리해보려는 시도에서 출발하였다. 식으로 표현하면 식 (3)과 같다.

$$\tan\Phi = 2^{-i} \tag{3}$$

식 (3)에 의해 식 (2)는 식 (4)와 같이 다시 정리할 수 있다.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \cos\Phi \begin{bmatrix} 1 & -2^{-i} \\ 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (4)$$

이와 같이 쉬프트 연산만으로 수행하는 회전을 의사 회전(pseudo rotation)이라 한다. 의사 회전은 초기 벡터에 대해 수직 방향으로 회전을 하게 되므로 회전 후 벡터의 크기가 늘어나게 된다. (그림 3)은 일반적인 회전과 의사 회전을 비교하여 나타낸 것이다.



(그림 2) 일반적 회전과 의사 회전

식 (4)에서 $\cos\Phi$를 제외하면 각 θ의 회전을 VLSI 회로로 구현하기 쉬운 쉬프트와 덧셈 연산만을 이용하여 구성할 수 있음을 알 수 있다.

그러나, 실제 일반적으로 사용되는 원형 모드의 회전을 하기 위해서는 식 (4)에서 회전 행렬의 앞에 따로 곱해진 $\cos\Phi$를 처리해 주어야 한다. 즉, 이 $\cos\Phi$가 초기의 원점으로부터 (x,y)에 이르는 거리, 즉 (x,y)를 종점으로 하는 벡터의 크기를 일정하게 유지시키는 효과를 내는 요소라고 할 수 있다. CORDIC 알고리즘에서는 의사 회전을 통해 목표각 만큼의 회전을 수행한 후에 일괄적으로 $\cos\Phi$의 효과를 적용하기 위해 의사 회전에 의한 기본각 회전이 끝난 후 스케일 팩터 보상(Scale Factor Compensation) 또는 스케일링 단계를 수행한다.

2.2 기본 CORDIC 알고리즘

회전 목표각이 θ일 때, 기본 CORDIC 알고리즘에서 θ는 식 (5)과 같이 기본각(elementary angle)들의 합으로 표현할 수 있다.

$$\theta = \sum_{i=0}^{N-1} \mu(i)e(i) + \xi_m \tag{5}$$

기본각 $e(i)$는 CORDIC의 원리에 의해 식 (6)과 같이 나타낼 수 있으며, 이를 통해 실제 계산된 기본각들을 (표 1)에 정리하였다.

$$e(i) = \tan^{-1}2^{-i} \tag{6}$$

(표 1) CORDIC의 기본각

i	계산식	기본각(rad)
0	$e(0) = \tan^{-1}(2^{-0})$	0.7854
1	$e(1) = \tan^{-1}(2^{-1})$	0.4636
2	$e(2) = \tan^{-1}(2^{-2})$	0.2450
3	$e(3) = \tan^{-1}(2^{-3})$	0.1244
4	$e(4) = \tan^{-1}(2^{-4})$	0.0624
5	$e(5) = \tan^{-1}(2^{-5})$	0.0312
6	$e(6) = \tan^{-1}(2^{-6})$	0.0156
7	$e(7) = \tan^{-1}(2^{-7})$	0.0078

식 (5)에서 N은 반복횟수를 나타내며, 이는 기본각의 개수와 동일하다. $\mu(i)$는 각도의 회전 방향을 나타내는 변수이고 각 반복에서 현재 위치와 목표각의 위치를 비교하여 +1 또는 -1로 결정된다. 이와 같이 목표각을 양 또는 음의 CORDIC 기본각들의 합으로 표현하는 것을 각 양자화(Quantization)라고 한다. ξ_m은 양자화 과정 중에서 부분각을 회전하면서 실제 목표각인 θ와 차이인 각 양자화 오차를 뜻한다. 여기서 N_A는 부분각의 총 개수이다. N_A번 까지의 기본각 회전을 수행한 후 누적된 부분각이 θ_i라 할 때, ξ_m은 식 (7)과 같다.

$$\xi_m \triangleq \theta - \sum_{i=0}^{N_A-1} \theta_i \tag{7}$$

(표 2)는 기본 CORDIC 알고리즘에서 원형 모드에 대한 기본각 회전과 스케일링을 요약한 것이다.

(표 2) CORDIC의 기본각 회전과 스케일링 요약

초기화:

$x(0),\ y(0),\ z(0)$

반복식:

For i = 0 to N - 1

$$\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} = \begin{bmatrix} 1 & -\mu(i)2^{-i} \\ \mu(i)2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$$

$$a(i+1) = a(i) - \mu(i)e(i),\ e(i) = \tan^{-1}(2^{-i})$$

End

스케일 보상:

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = K \begin{bmatrix} x(N) \\ y(N) \end{bmatrix}$$

$$= \left(\prod_{i=0}^{N-1} \sqrt{1+2^{-2i}} \right)^{-1} \begin{bmatrix} x(N) \\ y(N) \end{bmatrix}$$

2.3 기본 CORDIC을 개선한 오프라인 알고리즘

기본 CORDIC 알고리즘은 정밀도에 따라 사용하고자 하는 기본각의 개수가 결정된다. 이에 따라 전체 반복횟수 'N'이 주어지면 목표각이 무엇이든 상관없이 그만큼의 반복을 빠짐없이 순차적으로 수행해야 하기 때문에 비효율적인 상황이 발생할 수 있다.

예를 들어, 목표각이 90°인 경우를 생각해보면 첫 번째 기본각인 $\tan^{-1}2^{-0} = 45^{o}$를 두 번 양(+)의 방향으로 회전하고 나머지 반복횟수를 생략하게 되면 한치의 오차도 없이 정확히 도달하게 된다. 그러나 기본 CORDIC을 사용하면 (표 1)에 나와있는 첫 번째 기본각부터 N 번째 기본각까지 순차적으로 모두 사용해야 하기 때문에 오히려 좋지 않은 성능을 가져오게 된다.

이처럼 목표각을 사전에 알고있는 경우, 해당 목표각에 적합한 기본각 시퀀스와 방향에 관한 시퀀스를 찾아서 회전시킴으로써 연산 횟수를 줄이기 위해 오프라인 개선 알고리즘들이 제안되었다.

오프라인 알고리즘은 실시간으로 들어오는 임의의 입력에 대한 처리를 제외한, 사전에 입력을 알 수 있는 경우에 대해 주어진 목표각에 대한 회전을 수행할 때 오프라인 상의 처리를 통해 최소의 반복 연산만으로 회전을 효율적으로 수행 할 수 있는 방법이다. 이러한 오프라인 단계의 처리를 통해 기본 CORDIC을 개선한 알고리즘으로 MVR(Modified Vector Rotational)-CORDIC과 MSR(Mixed Scaling Rotation)-CORDIC이 제안되었다[12],[13].

MVR-CORDIC은 기본각을 반복해서 사용하거나 생략할 수 있도록 알고리즘을 수정하여 회전 시에 필요한 연산의 반복횟수를 감소시켰다. 이와 마찬가지로 오프라인 형태로 제안된 알고리즘인 MSR-CORDIC은 회전식 내에 스케일링 과정을 포함하여 회전과 동시에 스케일링을 처리할 수 있도록 하였다.

기존의 CORDIC 알고리즘에서는 기본각 회전이 끝난 후에 벡터의 크기가 일반적으로 초기 크기보다 크지만, MSR-CORDIC에서 회전시키는 벡터의 크기는 반복수행동안 초기 벡터 크기보다 작은 값을 가질 수 있다. 스케일 보상 단계과 회전

연산을 혼합하여 별도의 스케일링 연산이 필요하지 않도록 함으로써 스케일링 연산의 과부하 없이 반복 횟수는 줄이고 연산 속도는 올릴 수 있게 되었다.

본 논문에서는 2차원 CORDIC 알고리즘을 개선한 이러한 오프라인 기법을 기반으로 3차원 CORDIC-3D 알고리즘을 개선할 수 있는 오프라인 알고리즘을 제안하고자 한다.

3. CORDIC-3D 회전 알고리즘

CORDIC-3D 알고리즘은 주로 2차원 벡터회전을 수행하던 기본 CORDIC을 3차원 응용을 위해 확장한 것이다. Lang등은 3D 컴퓨터 그래픽스의 지오메트리 연산에서도 주된 변환 과정에 CORDIC 알고리즘을 적용함으로써 성능 향상을 기대할 수 있을 것으로 보고 3차원 벡터회전을 위한 CORDIC-3D 알고리즘을 제안하였다. 이 연구를 통해 데이터 집중적인 연산을 CORDIC형의 프리미티브로 연결하여, CORDIC 프로세서의 배열로 이루어진 하나의 프로세서 내에서 연산을 하면 컴퓨터 그래픽스에서의 다양하고 복잡한 연산의 결과를 얻을 수 있었다.

CORDIC-3D는 벡터링 모드와 벡터링 후 회전 모드로 구성된다. 벡터링 연산은 하나의 3차원 벡터를 두 개의 회전축 집합에서 기본 회전을 이용하여 주축의 하나로 회전한다. 일반적으로 x축을 주축으로 한다. 벡터링 후 회전 연산은 벡터링 연산에서 사용된 3차원 회전을 임의의 벡터들에 적용한다. CORDIC-3D의 알고리즘은 (그림 3)와 같다[14].

벡터의 초기 위치인 (x_0,y_0,z_0)에서 시작하여 반복식에 의해 전체 반복 횟수인 N만큼 회전을 수행하게 되는데, 벡터링에서 회전의 방향 $s_{1,j}$,$s_{2,j}$는 각각 x와 y좌표의 곱의 부호, x와 z좌표의 곱의 부호로 결정된다. 2차원 CORDIC 알고리즘에서와 마찬가지로 회전 후에 변경된 벡터의 크기를 복구시켜 주기 위해 스케일 팩터 보상 단계를 거친다. 이 스케일 팩터 보상 과정에서 사용되는 스케일 팩터를 K_{3D}로 나타내었다. 벡터링 단계의 결과로 회전에 사용하게 될 시퀀스 값이 결정되고, 벡터링 후 회전에서 미리 계산된 시퀀스를 이용해서 실제 회전을 수행하게 된다.

x_f는 회전 후 x좌표에 스케일 팩터를 곱한 것으로, 스케일링 과정 후에 벡터의 x좌표를 말한다. 따라서, x축을 기준으로 벡터링을 수행한 후에 x_f는 벡터링 후 해당 벡터의 크기와 일치한다.

1) **초기화** : (x[1],y[1],z[1]) = (x_0,y_0,z_0)
2) **반복식** :
For j = 1 to N
If (벡터링 모드) then // [벡터링] 연산인 경우
$(s_{1,j}, s_{2,j})$= (sign(x[j])sign(y[j]), sign(x[j])sign(z[j]))
else // [벡터링 후 회전] 연산인 경우
precomputed s_1, s_2
Endif

$$x[j+1] = x[j] - 2^{-2j+1}x[j] + s_{1,j}2^{-j+1}y[j] + s_{2,j}2^{-j+1}z[j]$$
$$y[j+1] = y[j] - s_{1,j}2^{-j+1}x[j] - s_{1,j}s_{2,j}2^{-2j+1}z[j]$$
$$z[j+1] = z[j] - s_{2,j}2^{-j+1}x[j] - s_{1,j}s_{2,j}2^{-2j+1}y[j]$$

Endfor
3) **스케일 보상**

$$x_f = K_{3D}x[N+1]$$
$$(K_{3D} = \Pi_{i=1}^{N} 1/(1+2^{-2i+1}))$$

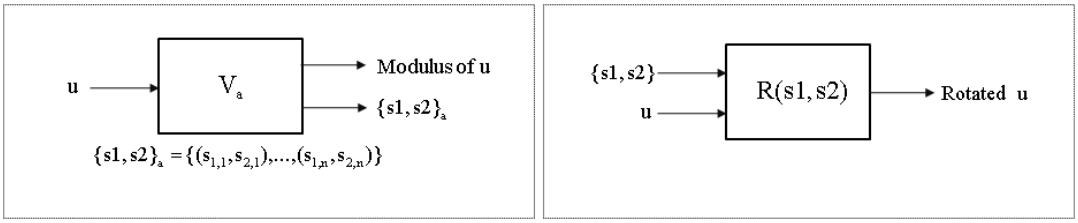
4) **결과(벡터링)**

$$x_f = \sqrt{x_o^2 + y_0^2 + z_0^2}, \{s_1, s_2\}$$

5) **결과(벡터링 후 회전)**
계산된 시퀀스 $\{s_1, s_2\}$를 이용하여 $(x[1], y[1], z[1])$를 회전한 좌표

(그림 3) CORDIC-3D 알고리즘

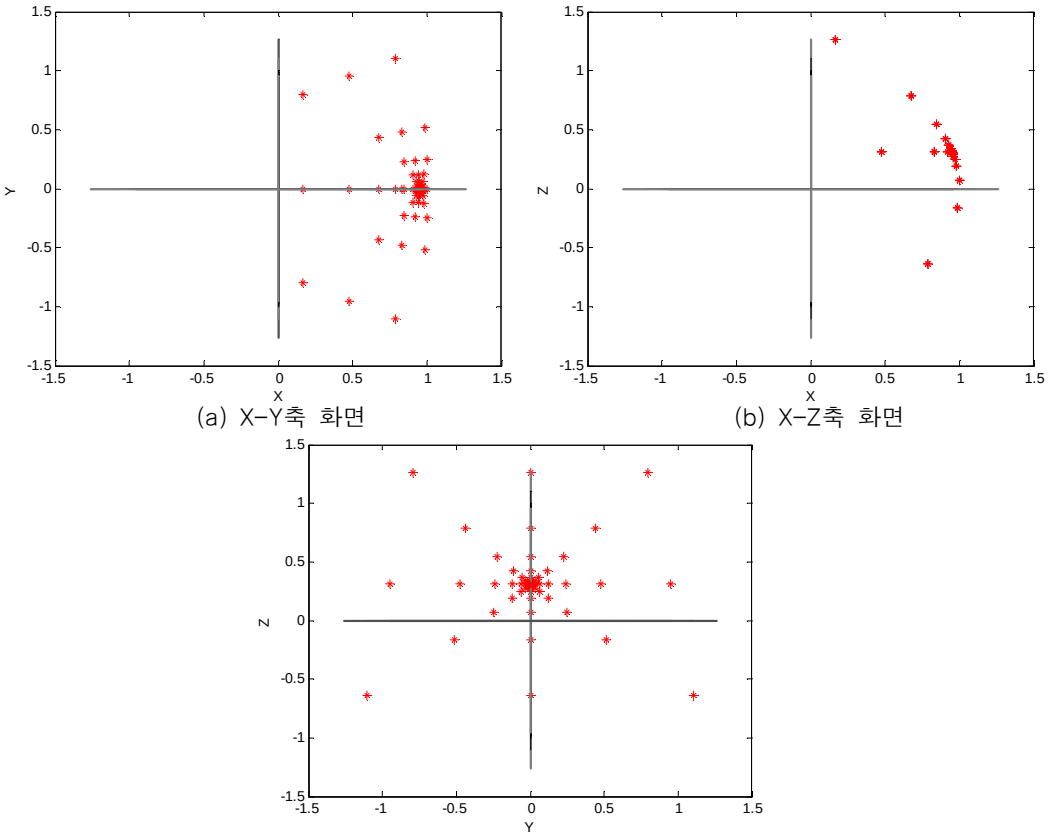
(그림 4)의 (a)와 (b)는 CORDIC-3D 연산자들의 표현을 나타내며, V_a과 $R(s1, s2)$의 두 가지 모듈로 구성된다[1].



(그림 4) CORDIC-3D 모듈
(a) 벡터링(Axis a) (b)회전(시퀀스 s1,s2를 이용)

V_a 연산자는 벡터 u의 회전을 수행하여 결과 벡터가 'α'축에 평행하도록 한다. 즉, 이 회전은 'α'축에 해당하는 것을 제외한 모든 좌표값을 0으로 만든다. 결과값이 0이 아닌 좌표는 벡터계수이며 벡터링 모드에서의 CORDIC-3D로 수행될 수 있다. $R(s1,s2)$는 벡터링 후 회전연산을 수행한다. 즉, V_a 연산자로 얻어진 방향 시퀀스 $\{s1,s2\}$를 이용하는 벡터 u의 회전이다. 이는 미리 계산된 시퀀스 $\{s1,s2\}$를 이용하여 CORDIC-3D 모듈로 수행한다.

(그림 5)은 3차원 상의 한 점 (0.9511, 0, 0.3090)을 초기 위치로 했을 때 한번의 CORDIC-3D 회전으로 도달 가능한 위치들을 표시한 것이다.

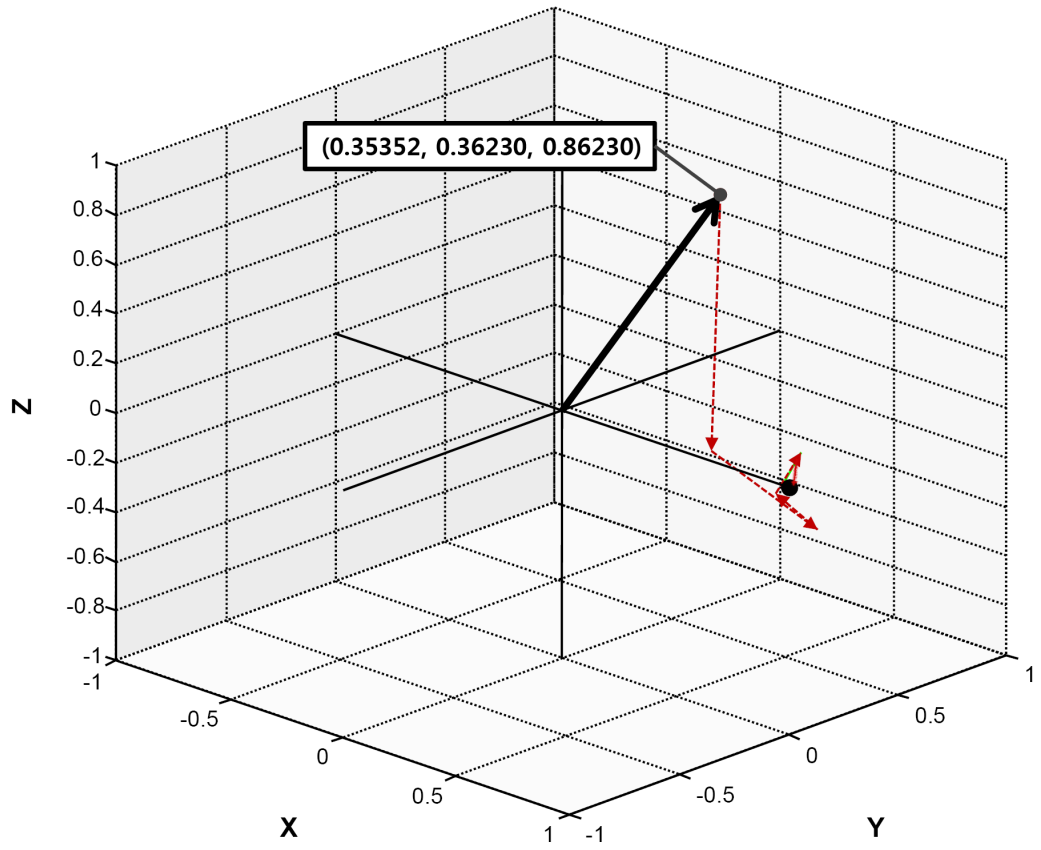


(a) X-Y축 화면 (b) X-Z축 화면

(c) Y-Z축 화면

(그림 5) CORDIC-3D가 한번 회전으로 도달 가능한 위치(N=8)

(그림 6)은 입력좌표 UI가 (0.35352, 0.36230, 0.86230)인 경우 V_x 모듈의 연산 결과를 보여준다. (그림 6)에서 UI에서 출발하여 벡터링을 수행하는 과정을 점선 화살표로 각 반복단계마다 나타내었다. UI 벡터의 좌표를 (x_0, y_0, z_0)라고 할 때, y 축과 z축에서 x 축 방향으로 회전을 하면서 벡터링 연산을 수행한다. 따라서 y_0와 z_0는 0으로 수렴한다.



(그림 6) V_x의 벡터링 결과 (UI = (0.35352, 0.36230, 0.86230))

Ⅲ. 개선된 *3D CORDIC* 회전 알고리즘의 제안

2차원 기본 CORDIC 알고리즘은 정밀도에 의해 결정되는 기본각의 개수 'N'을 전체 반복횟수로 가지며, 목표각에 상관없이 그만큼의 회전을 모두 수행해야 하므로 비효율적일 수 있었다. 이러한 단점은 기본 CORDIC을 3차원으로 확장시킨 CORDIC-3D 알고리즘에서도 그대로 이어졌다. 앞서 살펴본 CORDIC-3D는 기본 CORDIC 알고리즘을 기반으로 했기 때문에 입력의 특성을 고려하지 않고 모든 입력에 대해 'N'번의 기본각 회전을 수행해야 한다.

본 논문에서는 2차원 CORDIC을 개선시킨 오프라인 알고리즘에서 사전에 입력을 알고있는 경우 오프라인 상에서 입력에 맞는 시퀀스를 찾아서 회전하도록 하였던 것과 동일한 방식으로, 사전에 주어진 입력 3차원 좌표에 대한 최적의 시퀀스를 검색하고 이를 이용하여 회전하는 개선된 CORDIC-3D 알고리즘을 제안한다.

제안 알고리즘은 오프라인 벡터링 단계와 검색한 시퀀스로 실제 회전을 수행하는 벡터링 후 회전단계로 구성된다. 오프라인 벡터링 단계에서 각도 기반 검색과 단위벡터 기반으로 스케일링을 고려한 검색의 두 가지를 제시하고 이를 통해 최적의 시퀀스를 결정하여 CORDIC-3D의 반복횟수를 줄이고자 하였다.

1. 오프라인 벡터링

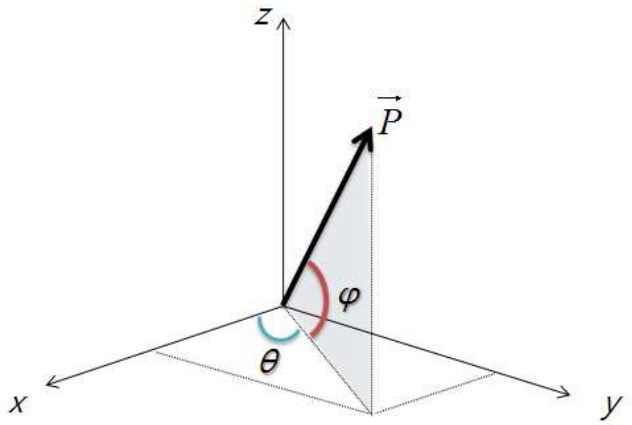
1.1 각도 기반 검색 알고리즘(Angle Based Search-3D : ABS-3D)

3차원 벡터 표현에서 축과 각 표현법에 따르면 3차원 상의 임의의 벡터는 두 개의 각으로써 표현될 수 있다. 이에 착안하여 (그림 7)과 같이 3차원 벡터 P를 θ와 ϕ의 두 각으로 표현하고 이를 검색의 기준으로 활용하였다. θ는 벡터 P를 x-y 평면상으로 직교 투영시킬 때 x축과 이루는 각을 나타내고, ϕ는 이때의 수직면상에 생기는 즉, x-y평면과 벡터 P 사이에 형성되는 각을 의미한다.

벡터링을 수행하기 위해서는 벡터 P가 결과적으로 x축에 접하도록 만들어야 하기 때문에 두 각이 0에 수렴하도록 해야 한다. 이러한 내용을 기준으로 시퀀스를 결정하도록 하였다.

검색을 위한 오차성능의 지표로서 e_θ,e_ϕ를 각각 θ와 ϕ의 절대값, $|\theta|$,$|\phi|$으로 정의하고 각 θ는 실수좌표 x와 y로 이루어지는 4사분면상의 역탄젠트 값으로 계산될 수 있으며, ϕ는 z와 $\sqrt{x^2+y^2}$ 사이에 이루어지는 역탄젠트 값으로 계산할 수 있다.

본 논문에서는 해당 조건에 가장 적절한 방향 및 회전 시퀀스를 찾아내기 위한 최적화 알고리즘으로 그리디 탐색(greedy search)법과 완전 탐색(exhaustive search)법을 적용하였다.



(그림 7) 두 각에 의한 3차원 벡터의 표현

[그리디 탐색]

그리디 탐색은 현재의 반복단계에서 최적의 시퀀스로 결정되는 지역적 최적해를 통해 전체의 최적해를 이끌어내는 방식으로 최적화 문제를 간단하고 빠르게 해결할 수 있다. 반복횟수 N′에 따라 각 단계에서 가장 낮은 오차값을 갖는 방향 시퀀스 d_1,d_2와 기본각 시퀀스 s를 검색하여 저장한다.

검색과정에서 오차 수치의 최소값을 저장하고 비교 조건으로 이용하기 위해 변

수 $e_{\min}$를 사용하였다. 기본 CORDIC-3D 알고리즘과 달리 MVR-CORDIC의 아이디어에 기반하여 방향 시퀀스에 0이 추가 되었으며 이는 최적의 시퀀스 결정을 위해 불필요한 시퀀스를 통한 회전을 생략하는 과정에서 필요하다.

초기화 단계에서 반복횟수를 나타내는 변수 i와 성능 비교를 위한 최소값 저장 변수 $e_{\min}$를 각각 1과 충분히 큰 수로 초기화 한다. 입력좌표 UI와 워드길이, 전체 반복횟수 등은 사전에 주어진다. 입력좌표로 주어지는 목표 위치를 3차원 CORDIC 회전 연산으로 도달하기 위해서는 방향시퀀스 $d_{1,}d_2$와 회전시퀀스 s를 결정해야 한다. 방향 시퀀스는 음의 방향, 양의 방향 또는 기본각 회전 생략의 세 가지 중 하나로 결정되며, 기본각 생략은 추가 회전이 불필요한 경우 현 위치를 유지할 수 있도록 하는 시퀀스이다[15]. 회전시퀀스는 전체 기본각 집합 중 해당 반복에서 몇 번째 기본각을 사용할 것인가를 나타내며, 기본각의 개수는 워드길이에 따라 결정된다.

방향시퀀스와 회전시퀀스를 조합하여 각 경우에 대한 회전 수행 후의 오차값 $e_\theta(i)$, $e_\phi(i)$를 계산하고 전 단계까지의 최소값이 저장되어 있는 $e_{\min}$과 비교한다. 새로 계산된 오차값이 더 작은 경우에는 $e_{\min}$을 갱신함으로써 해당 반복에서 가장 작은 오차값을 만족시키는 경우의 시퀀스를 찾을 수 있다. 그리디 알고리즘의 경우 반복에 따라 순차적으로 시퀀스를 결정하기 때문에 초기에 큰 기본각들이 사용되고 후반으로 갈수록 미세한 조정을 위해 작은 각들이 사용되는 특성을 갖는다.

(그림 8)는 그리디 탐색 알고리즘을 정리한 것이다.

1) 초기화: $i=1$, $e_{\min}= \infty$
2) 초기 입력좌표(벡터) UI, 워드길이 W, 반복횟수 N'이 주어짐
3) $d_1(i),d_2(i)\in\{-1,0,1\}$, $s(i)\in\{1,2,3,\cdots W\}$일 때, 각 경우에 회전 후의 UI'(x(i+1),y(i+1),z(i+1))와 오차값 $e_\theta(i),e_\phi(i)$를 계산
4) 오차값 $e_\theta(i),e_\phi(i)$를 전 단계의 오차 최소값 $e_{\min}$과 비교하여 값이 더 작으면 d1(i),d2(i),s(i)를 저장하고, $e_{\min}$ 값을 갱신(update)
5) i < N'이면 수행 종료, 그 외의 경우 i는 1증가하고 3)단계부터 재 수행

(그림 8) ABS-3D의 그리디 탐색 알고리즘

[완전 탐색]

각각의 반복 내에서 오차값을 최소화하는 시퀀스 조합을 결정하는 방식인 그리디 탐색과는 달리, 완전 탐색은 가능한 모든 경우를 다 고려하여 1부터 N까지의 전체 반복을 수행했을 때의 오차값을 측정하고 그 중 가장 조건에 알맞는 경우를 찾아내는 방법이다. 가질 수 있는 모든 시퀀스의 경우에 따라 각각의 오차를 계산하여 비교해야 하므로 그리디 탐색에 비해 검색에 소요되는 시간적인 비용이 상당히 크다는 단점이 있으나, 가능한 전체 해집합을 모두 검색하기 때문에 그리디 탐색법보다 전반적으로 결과의 정확도가 높다.

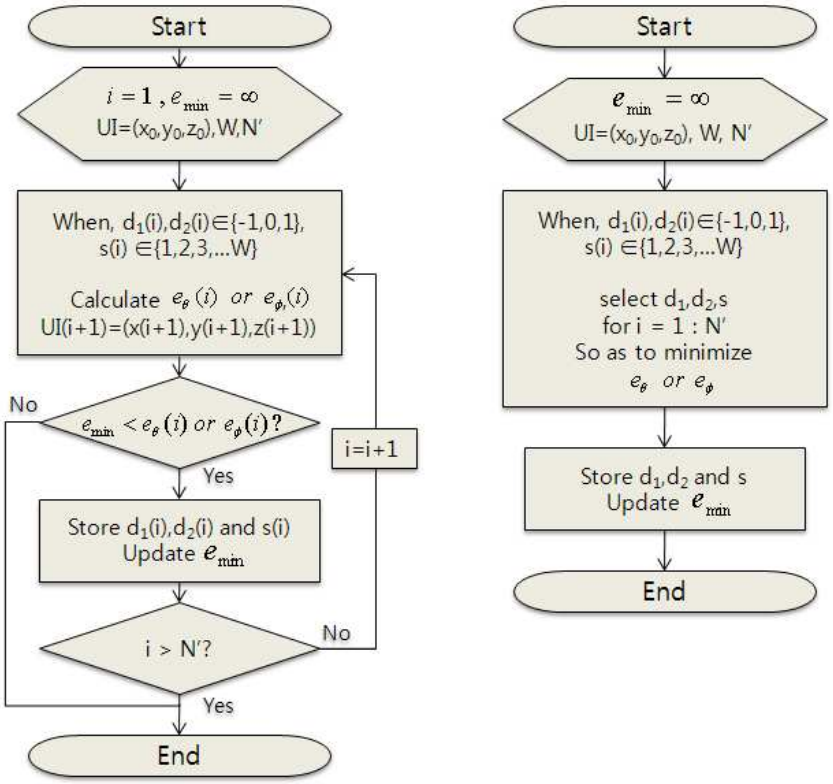
초기화 단계는 그리디 탐색과 동일하다. 완전 탐색의 경우 시퀀스 조합을 결정하기위해 전체 반복횟수인 N개 만큼의 방향시퀀스와 회전시퀀스 조합을 선택하고 이들을 이용해서 회전한 최종 좌표에서의 오차값 e_θ,e_ϕ를 계산한다. 모든 경우를 탐색하여 최소의 오차값을 만족하는 전체 시퀀스 조합을 결정하는 방식이다.

(그림 9)은 완전 탐색 알고리즘을 정리한 것이다.

1) 초기화: $i=1$, $e_{\min}$= ∞
2) 초기 입력좌표(벡터) UI, 워드길이 W, 반복횟수 N'이 주어짐
3) $d_1(i),d_2(i)\in\{-1,0,1\}$, $s(i)\in\{1,2,3,\cdots W\}$일 때, 시퀀스 집합 d_1,d_2,s를 선택하고 반복횟수 N' 전체에 대한 회전 후의 UI'(x,y,z)와 오차값 e_θ,e_ϕ를 계산
4) 오차값 e_θ,e_ϕ를 전 단계의 오차 최소값 $e_{\min}$과 비교하여 값이 더 작으면 시퀀스 집합 d_1,d_2,s를 저장하고, $e_{\min}$ 값을 갱신(update)
5) i < N'이면 수행 종료, 그 외의 경우 i는 1증가하고 3)단계부터 재 수행

(그림 9) ABS-3D의 완전 탐색 알고리즘

위에서 설명한 두 가지 탐색 알고리즘은 (그림 10)의 순서도로 표현할 수 있다.



(그림 10) ABS-3D의 그리디 탐색, 완전 탐색의 순서도

(표 3)는 입력좌표 UI=(0.8232, 0.4194, 0.3827)인 경우에 그리디 탐색 기반 알고리즘으로 결정된 방향과 회전 시퀀스 집합을 정리한 것이다. 기본 CORDIC-3D 알고리즘의 경우와는 달리 각 단계에서 최적의 오차조건을 만족하지 못하는 특정 회전 시퀀스를 생략하고 그보다 나은 것을 적용함으로써 효율적인 회전이 가능하다.

(표 3) UI=(0.8232, 0.4194, 0.3827)일 때, ABS-3D의 시퀀스 집합 d1,d2,s(N'=4)

i	*1*	*2*	*3*	*4*
$d_1(i)$	1	-1	1	-1
$d_2(i)$	1	-1	0	-1
s(i)	2	5	6	8

1.2 스케일링 효과를 고려한 검색 알고리즘(Scaling Considered Search-3D :SCS-3D)

여기에서는 시퀀스 검색의 기준을 벡터링을 수행한 후에 입력벡터가 x축에 접하는 단위벡터 $v(1,0,0)$로 수렴하도록 한다. 이는 벡터링 과정에서 결과 벡터의 크기까지 고려하여 최대한 단위 길이로 유지할 수 있는 시퀀스들을 선택하도록 함으로써 스케일링의 오버헤드를 줄일 수 있다.

여분의 스케일링 과정이 필요하지 않는다는 점에서 MSR-CORDIC과 유사하지만, MSR-CORDIC이 회전식 내에 스케일링 연산식을 포함시키는 형태인 것과는 달리 SCS-3D는 시퀀스를 검색하는 과정에서 스케일 팩터를 1에 가깝게 만드는 시퀀스를 골라서 결정되도록 하는 방식이다.

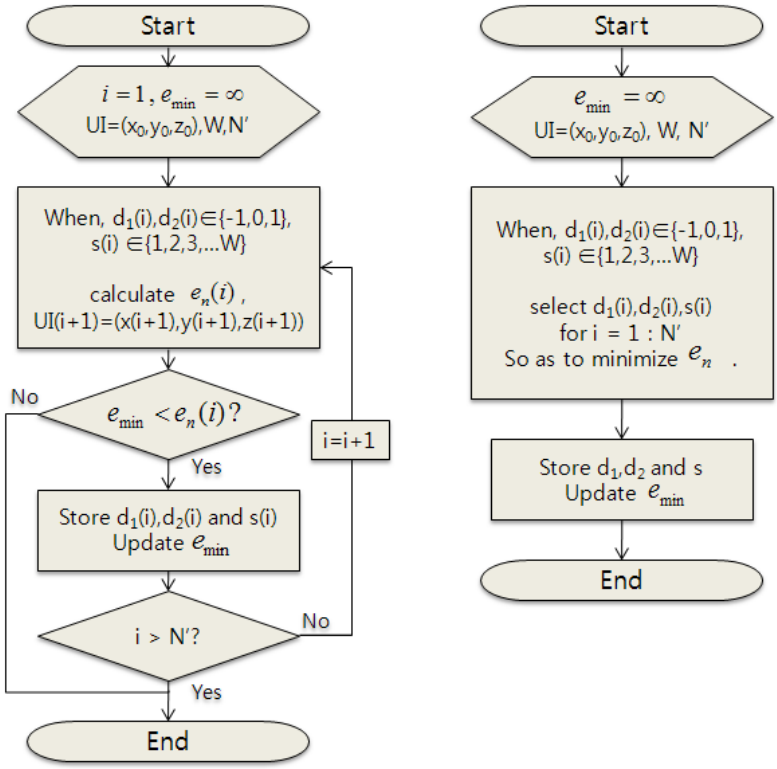
SCS-3D 알고리즘은 3장의 1.1에서 설명한 ABS-3D와 마찬가지로 그리디 탐색과 완전탐색을 이용하여 시퀀스를 검색한다. 탐색 과정은 ABS-3D에서와 거의 유사하나 시퀀스 선택의 기준이 변함과 동시에 오차성능의 지표도 달라졌다.

스케일링 효과를 고려한 알고리즘에서의 오차성능의 지표로 사용된 e_n는 다음과 같이 계산된다.

$$e_n = n_d - n_f \tag{9}$$

$$n_{d=}\sqrt{x_d^2+y_d^2+z_d^2},\ n_f = x(i+1) \tag{10}$$

n_d는 회전 후 벡터의 이상적인 크기를 나타내며, 본 알고리즘의 벡터링 단계에서는 단위벡터에 접근하도록 하였으므로 n_d는 1이 된다. 이 두 가지 탐색의 순서도는 (그림 11)와 같다.

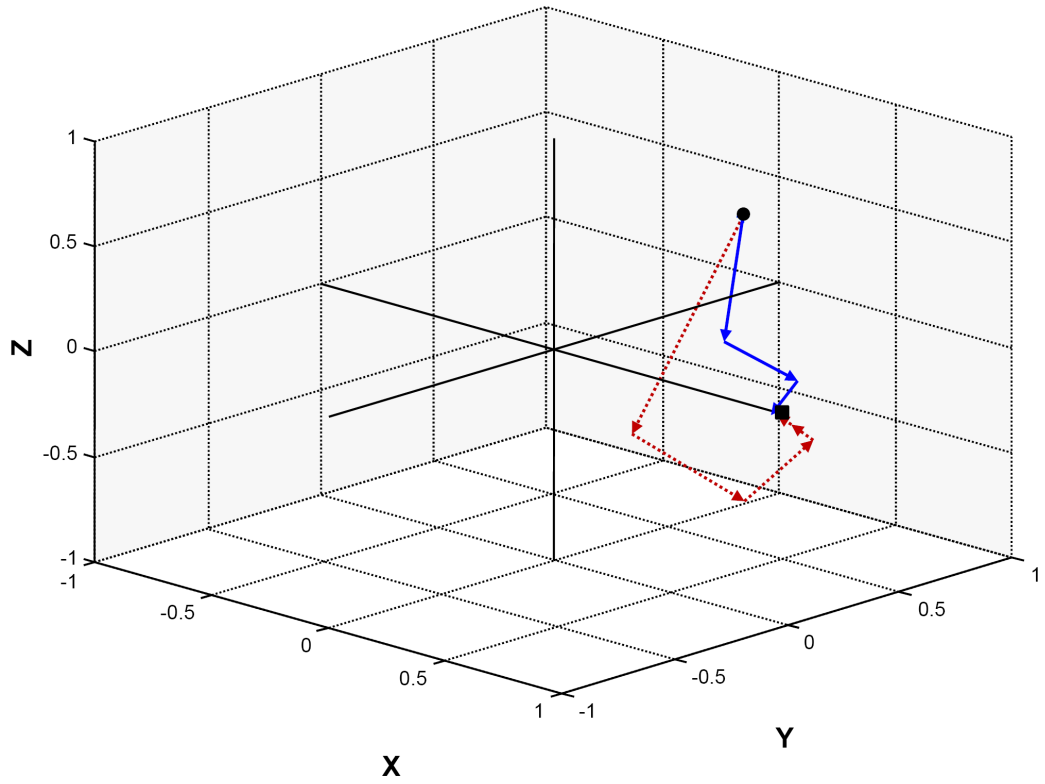


(그림 11) SCS-3D의 그리디 탐색과 완전 탐색의 순서도

(표 4)은 입력값 UI=(0.8232, 0.4194, 0.3827)인 경우의 방향과 회전 시퀀스 집합을 정리한 것이다.

(표 4) UI=(0.8232, 0.4194, 0.3827)일 때, SCS-3D의 시퀀스 집합 d1,d2,s (N'=4)

i	*1*	*2*	*3*	*4*
$d_1(i)$	1	-1	1	1
$d_2(i)$	1	-1	-1	1
$s(i)$	2	5	7	8



(그림 12) CORDIC-3D와 SCS-3D의 벡터링 비교 (N=6, N'=4)

(그림 12)는 원형점으로 표시된 (0.6177,0.2007,0.7604)를 입력좌표로 하여 두 알고리즘의 벡터링 경로를 나타낸 것이다. 점선으로 표시된 경로가 CORDIC-3D에 의한 것이고, 실선으로 표시된 경로가 본 논문에서 제안된 SCS-3D 알고리즘에 의한 벡터링 과정을 나타낸다. 기본 CORDIC-3D는 여러 번의 반복을 통해 점차적으로 벡터링이 이루어지기 때문에 이동경로가 크고 그만큼 스케일링의 오버헤드도 증가하게 된다. 반면에 제안된 알고리즘에 의한 벡터링은 보다 적은 반복횟수로 빠르게 목표지점에 도달할 수 있다. 또한 스케일링 효과를 고려하여 시퀀스가 결정되었기 때문에 별도의 스케일링이 필요하지 않게 된다.

2. 벡터링 후 회전단계

본 단계에서는 오프라인 벡터링 단계에서 최적화 알고리즘을 통한 시퀀스 검색으로 결정되는 방향시퀀스 d_1 ,d_2와 회전시퀀스 s를 이용하여 실질적인 3차원 벡터 회전 연산을 수행하게 된다. 여기서 d_1 ,d_2 s는 각각 반복 단계마다 사용될 방향 정보와 기본각 정보를 가진 시퀀스 집합이다.

1) **초기화**: x[1],y[1],z[1]=(x_0,y_0,z_0)

2) **반복식**

For j =1 to N'

$$x[j+1] = x[j] - 2^{-2s(j)+1}x[j] + d_1(j)2^{-s(j)+1}y[j] + d_2(j)2^{-s(j)+1}z[j]$$

$$y[j+1] = y[j] - d_1(j)2^{-s(j)+1}x[j] - d_1(j)d_2(j)2^{-2s(j)+1}z[j]$$

$$z[j+1] = z[j] - d_2(j)2^{-s(j)+1}x[j] - d_1(j)d_2(j)2^{-2s(j)+1}y[j]$$

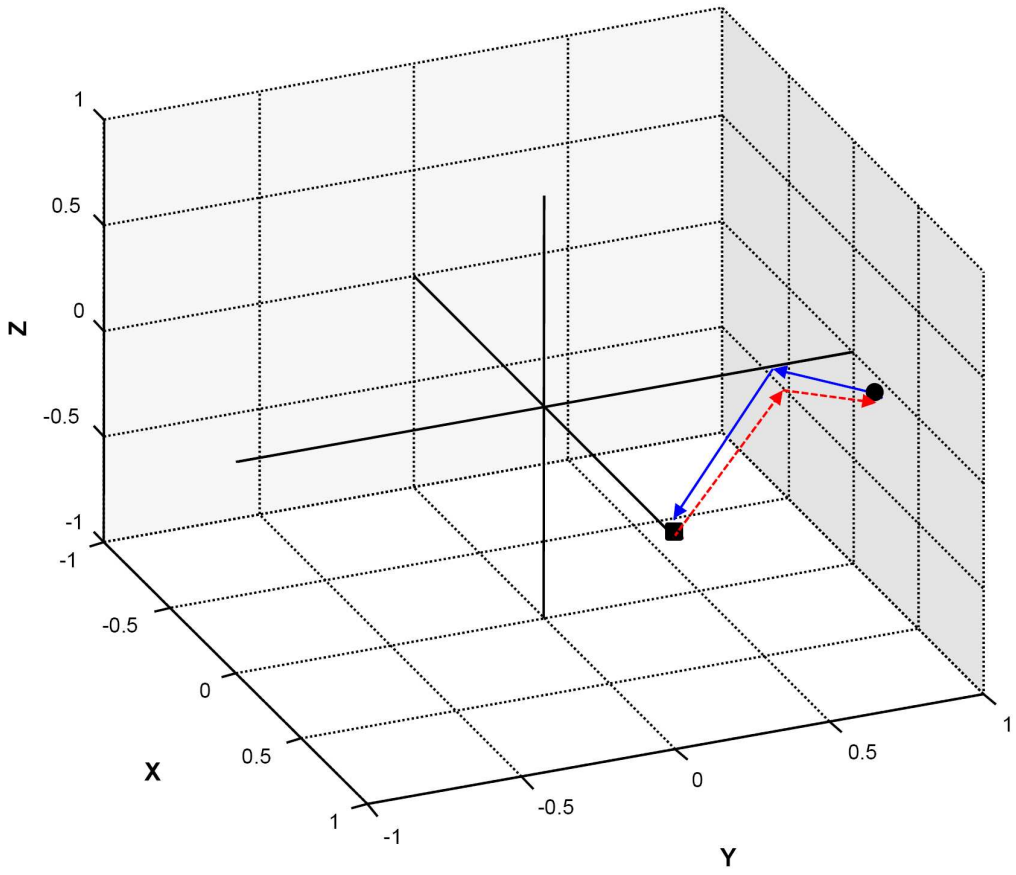
End

(그림 13) 제안된 CORDIC-3D 벡터링 후 회전 알고리즘

초기 입력 3차원 좌표, 즉 회전을 적용하게 될 대상 좌표를 초기화 하고 반복횟수마다 미리 계산된 시퀀스 정보들로 회전을 수행한다. 이때, 기본 CORDIC-3D 알고리즘에서와 같이 방향과 회전 시퀀스의 입력에 주의하여야 한다. 되돌아 오도록 회전하여야 하므로 방향 시퀀스의 경우 $\{-d_1,-d_2\}$와 같이 반대로 입력해야 하며, 회전 시퀀스 또한 입력 순서를 역으로 하여 $\{s(N'),s(N'-1),...,s(1)\}$과 같이 조정해 주어야 한다.

(그림 14)는 이러한 과정을 적용하여 벡터링 단계에서 결정된 시퀀스 집합을 이용하여 실제로 벡터링 후에 회전이 제대로 이루어 지는가를 확인해 본 그림이다. 원형의 점으로 표시된 것은 입력좌표 UI를, 사각형의 점은 벡터링 후 위치이자 회전의 시작점인 (1,0,0)이다. 실선 화살표로 표시된 경로가 입력좌표에서 출발하여

벡터링하는 과정이고, 점선 경로가 벡터링 후 회전의 과정이다. 정밀도를 어느 정도로 결정하는지에 따라 약간의 오차가 발생하지만 대체로 회전이 잘 수행됨을 확인할 수 있다.

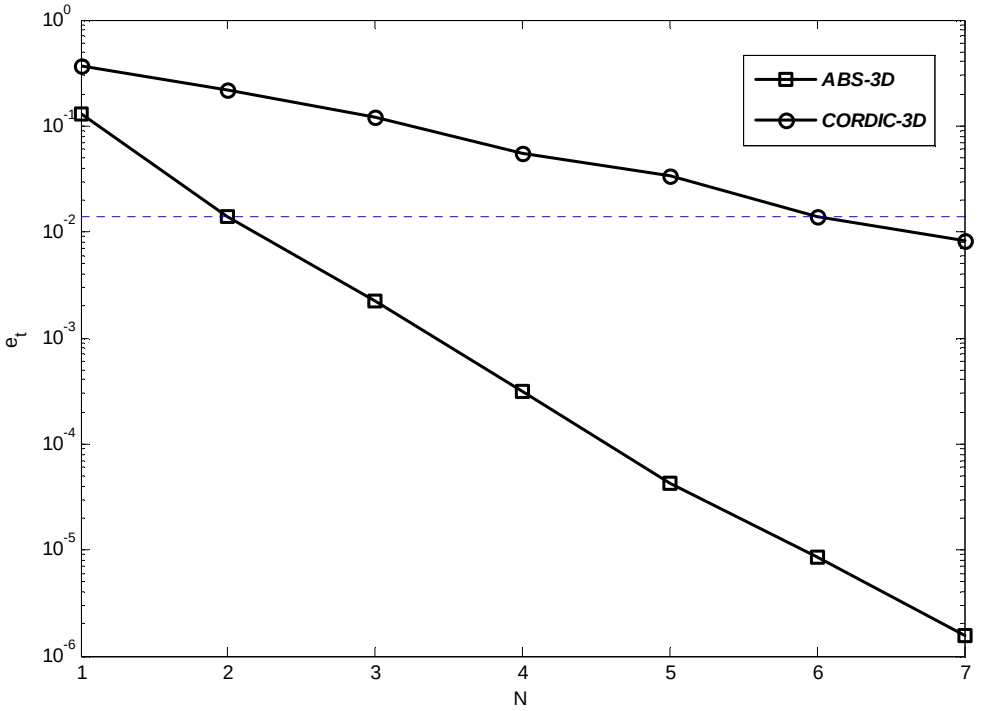


(그림 14) UI=(0.3090,0.9511,0)에 대한 벡터링과 회전의 경로

3. 실험결과

3.1 ABS-3D의 실험결과

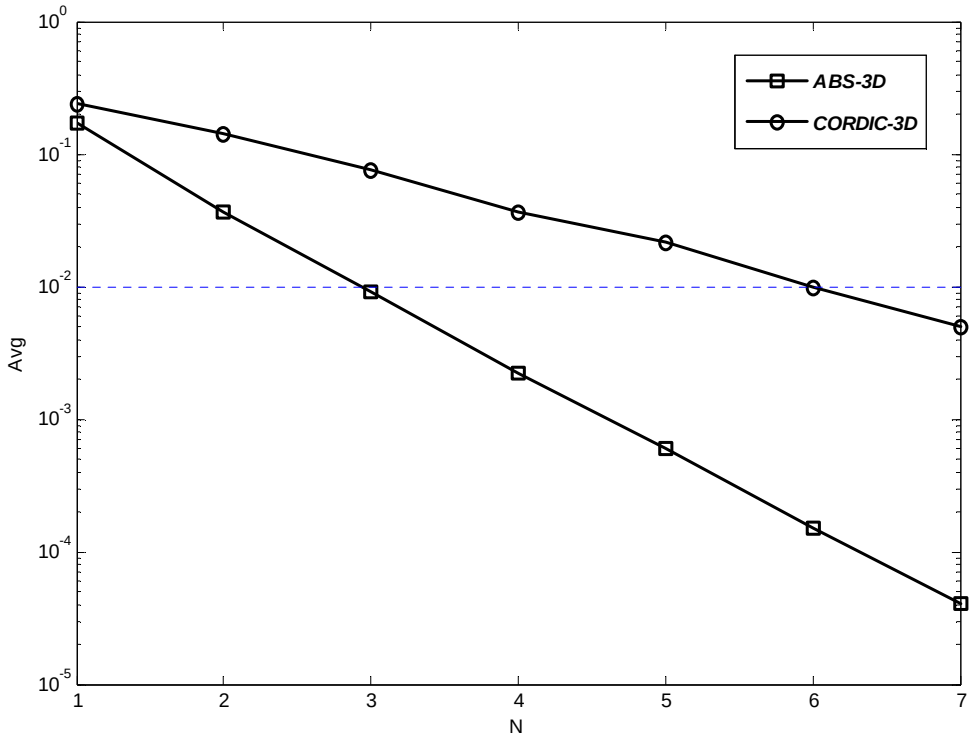
(그림 15)는 ABS-3D 알고리즘을 적용한 경우 그리디 탐색법을 통한 반복횟수의 변화에 따른 θ의 오차값 e_t의 변화를 그래프로 나타낸 것이다. 기본 CORDIC-3D 알고리즘의 오차값과의 비교를 위해 CORDIC-3D의 오차 그래프도 함께 표시하였다. 점선으로 표시된 지점에서 비교하여 보면, 동일한 오차값을 갖게 되는 반복횟수가 CORDIC-3D의 경우는 6번, ABS-3D의 경우 2번이라는 것을 확인할 수 있다. 이는 CORDIC-3D의 반복횟수가 N일 때, 제안된 ABS-3D는 1/3N의 반복만으로도 동일한 성능을 낼 수 있음을 보여준다.



(그림 15) 반복횟수 N에 따른 e_t의 변화

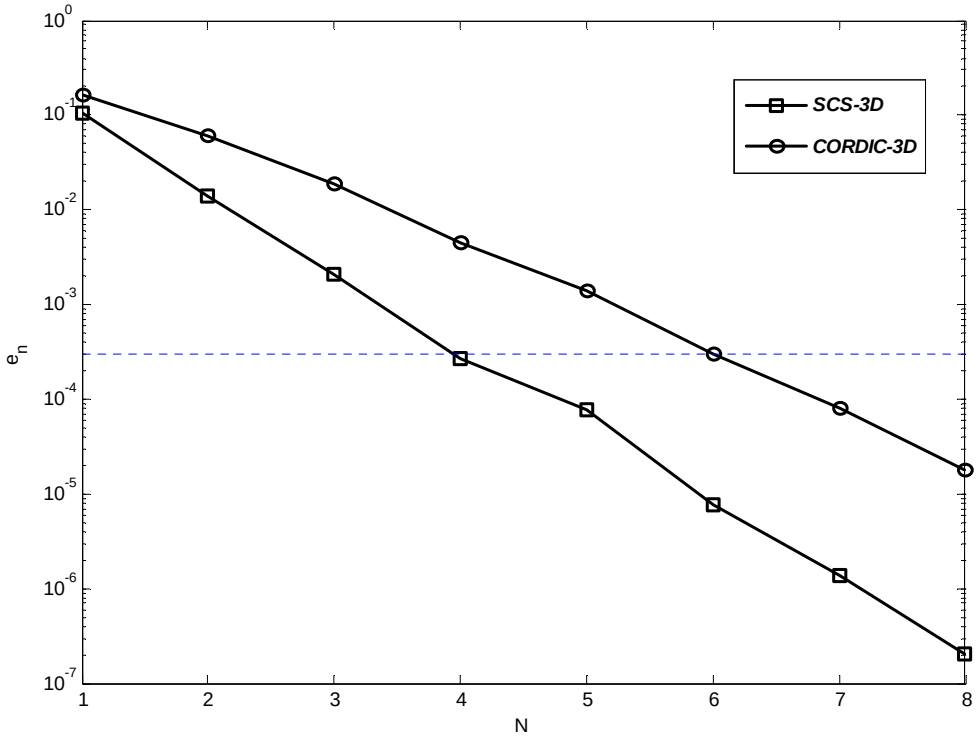
여기서, 추가적으로 고려해야 할 사항이 있는데, 바로 θ와 ϕ 중 어느 각에 대한 최적화에 초점을 둘 것인가의 문제이다.

(그림 16)은 θ와 ϕ의 오차인 e_t,e_p의 평균값을 기준으로 했을 경우, 평균 오차 Avg의 변화를 나타낸 것이다. θ와 ϕ 는 독립적이지 않고 서로의 값에 영향을 미치기 때문에 θ를 중심으로 한 시퀀스는 θ를 최적화 하는 문제에만 치중하게 되어 ϕ의 오차를 상승시키게 되고, 반대로 ϕ에 중점을 둔 시퀀스 검색은 θ의 정확도를 떨어뜨리게 된다. 따라서 평균 오차값을 기준으로 사용한 경우에는 (그림 15)의 경우에서 보다 전반적으로 오차값이 상승하는 효과가 나타나게 된다. 이러한 θ와 ϕ의 관계에 의한 효과를 좀 더 효율적으로 조절하기 위해 입력벡터의 특성에 따라 각각의 경우를 선택적으로 적용해 볼 수 있다.



(그림 16) 반복횟수 N에 따른 평균오차 Avg의 변화

3.2 SCS-3D의 실험결과



(그림 17) 반복횟수 N에 따른 e_n의 변화

(그림 17)은 제안된 SCS-3D에서의 반복횟수 N에 따른 오차값의 변화를 나타내고, CORDIC-3D 알고리즘의 오차 그래프를 함께 표시하여 비교하였다.

SCS-3D 알고리즘의 경우 검색과정 자체에 스케일링 효과가 적용되어 있기 때문에 적절한 비교를 위해서는 CORDIC-3D의 회전단계의 반복횟수 N과 함께 스케일링에 소요되는 반복횟수까지 함께 고려하여야 한다. CORDIC-3D에서 회전 단계와 스케일링 단계에서 소요되는 반복횟수의 비율을 CORDIC과 비슷한 수준으로 예상하고 평균 N:N/3정도 소요된다고 할 때[16], 전체 반복횟수는 4N/3으로 계산된다. SCS-3D의 경우는 2N/3정도의 반복만으로도 동일한 오차 정도를 낼 수 있다. 따라서 전체 반복횟수의 비가 4N/3 : 2N/3, 즉 2:1로 전체 회전에 소요되는 시간을 절반으로 감소시켜 계산 속도를 크게 향상 시킬 수 있다.

3.3 실험결과 고찰

앞서 살펴본 바와 같이 본 논문에서 제안된 오프라인 기반의 검색 알고리즘인 ABS-3D와 SCS-3D는 사전에 목표 위치가 주어진 경우에 대해 오프라인 벡터링 단계에서 최적의 시퀀스를 검색하여 이용함으로써 기존의 CORDIC-3D 알고리즘에 비해 훨씬 적은 수의 반복만으로 동일한 오차 성능을 달성할 수 있었다. (표 5)는 각 알고리즘의 반복연산의 횟수를 비교한 것이다.

(표 5) CORDIC-3D와 본 논문에서 제안된 알고리즘의 연산 횟수 비교

	CORDIC-3D [1]	**본 논문에서 제안된 알고리즘**	
		ABS-3D	*SCS-3D*
반복횟수	$\frac{4}{3}N$	$\frac{4}{9}N$	$\frac{2}{3}N$

최적 시퀀스 탐색에서 각도 기반의 검색을 하는 ABS-3D 알고리즘의 경우 기존의 CORDIC-3D 알고리즘과 비교하여 반복횟수를 1/3로 감소시킴으로써 수행 시간을 약 66.7% 감소시켰으며, 스케일링 효과를 고려한 SCS-3D의 경우에는 반복횟수가 절반으로 감소하면서 수행시간을 50% 감소시킬 수 있다.

이러한 계산 속도의 개선과 함께, 향후 전력소모 관점에서의 연구를 위해 Voltage Scaling 방법을 적용하여 전력 소모의 절감 효과도 예상해 볼 수 있다.

디지털 회로의 전력 소모 P는 "$C(V_{DD})^2f$"에 비례한다. 회로의 지연시간 T가 V_{DD}에 비례한다고 가정하자. 본 논문에서 제안한 두 알고리즘은 기존의 성능 지연 T를 줄여 각각 T/3, T/2로 만들었기 때문에, V_{DD}를 동일한 비율로 줄이며, CORDIC-3D 알고리즘과 유사한 성능을 같게 되는데, 이때의 전력 소모는 다음과 같이 기술할 수 있다.

ABS-3D의 경우 : $C(V_{DD}/3)^2f = 1/9*“C(V_{DD})^2f$

SCS-3D의 경우 : $C(V_{DD}/2)^2f = 1/4*“C(V_{DD})^2f$

이는 CORDIC-3D 알고리즘에 비해 같은 성능을 유지하면서, 기존 연산 방법의 전력 소모에 대해 약 1/9, 1/4만의 전력소모를 필요로 한다는 것을 의미한다. 이러한 점에서 볼 때, 고성능의 특성이 절대적이지 않은 응용분야에서 제안 알고리즘들을 통해 시스템을 설계함으로써 설계자는 전력 소모를 적절히 조절할 수 있게 되며 특정 응용분야에 맞는 요구 성능과 전력소모 요건에 맞도록 구성할 수 있다.

Ⅳ. 결론

3차원 그래픽 처리과정은 많은 계산량을 필요로 한다. 또한 최근의 3차원 그래픽 처리에서는 보다 현실감 있는 정교한 표현이 요구되기 때문에 고성능의 연산이 수행 가능한 3차원 지오메트리 연산 장치가 필수적이다.

그러나 휴대용 기기에서의 3차원 그래픽 처리는 이러한 고성능의 요구조건 뿐만 아니라 면적 비용과 전력 소모 등이 추가적으로 고려되어야 한다. 이러한 조건을 충족시키기 위해 CORDIC 알고리즘을 3차원 벡터 회전 연산에 적용하려는 연구가 이루어졌으나, 기존에 제안된 CORDIC-3D 알고리즘은 2차원 기본 CORDIC이 가지고 있던 비효율성의 문제점을 그대로 가지고 있었다. 본 논문에서는 CORDIC-3D 알고리즘을 개선하기 위해 2차원 CORDIC을 바탕으로 제안되었던 MVR-CORDIC, MSR-CORDIC과 같은 오프라인 검색 기법을 기반으로 ABS-3D와 SCS-3D 알고리즘을 제안하여 3차원 벡터회전 연산의 속도 성능을 개선하는 방법을 연구하였다.

이는 사전에 입력을 알 수 있는 특정한 경우에 한하여 각 입력에 대한 최적의 시퀀스를 검색하여 이를 통해 회전시킴으로써 반복횟수를 줄이면서 성능을 개선할 수 있었다.

CORDIC은 2차원 회전을 다루는 하드웨어 연산 알고리즘으로서 다양한 수학적 연산을 고성능으로 수행할 수 있으며, 특히 벡터회전을 기반으로 한 연산 분야에서 적용되어 왔다. 이러한 CORDIC은 2차원적 계산을 주로 수행하였으나 최근 주축을 중심으로 3차원 회전 확장이 가능하고, 이러한 확장을 통하여 3차원 그래픽의 기본 연산으로 CORDIC-3D 회전 알고리즘이 제시되었다.

CORDIC-3D 기본 연산은 벡터링 모드와 벡터링 후 회전 모드가 있다. 벡터링 연산은 하나의 벡터를 두 개의 회전축 집합에서 3차원 기본 회전을 이용하여 주축의 하나로 일치시킨다. 벡터링 후 회전 연산은 벡터링 연산에서 사용된 3차원 기본 회전이 다른 벡터들에 적용한다.

기존에 제시된 CORDIC-3D 연산 알고리즘에서는 회전각에 수렴하기 위하여 고

정된 일련의 회전각을 사용하고 있고, 정밀도에 의해 결정된 전체 기본각의 수만큼 회전을 모두 수행해야 하기 때문에 수행 속도 면에서 단점을 가졌다. 이에 본 논문에서는 모바일 기기에 적합한 지오메트리 연산의 개선을 위한 오프라인 알고리즘인 ABS-3D와 SCS-3D 알고리즘을 제안하고 성능을 평가하였다. 사전에 입력좌표가 주어진 경우, 입력의 특성에 따라 해당 조건을 만족하는 최적의 시퀀스를 검색하여 반복 횟수를 줄임으로써 연산량을 각각 66.7%, 50% 감소 시켰으며 향후 전력 소모에 감소에 관한 연구 가능성까지 살펴보았다.

본 연구를 토대로 휴대용 단말기기의 특성을 만족시키면서, 동시에 고성능 그래픽 처리 기능을 갖는 모바일 멀티미디어 시스템을 구성할 수 있을 것이다.

앞으로 보다 효율적이고 다양한 최적화 탐색 알고리즘의 적용을 통한 추가적인 연구가 필요하며, 실제 제안된 알고리즘을 하드웨어로 구현하여 면적 또는 전력 소모에 대한 효과를 검증해야 할 것이다. 그리고 본 논문에서 제안한 알고리즘을 실제 모바일용 지오메트리 프로세서에 적용하여 3차원 그래픽 프로세싱 유닛을 구성하고 3차원 영상에 대한 변환과 조명 등의 처리를 수행하여 기존의 지오메트리 연산 구조와 비교하여 평가해 볼 수 있을 것이다.

참고문헌

[1] Tomás Lang and Elisardo Antelo, “High-Throughput CORDIC-based Geometry Operations for 3D Computer Graphics", IEEE Trans. on Computers, Vol.54, No.3, pp.347~361, Mar. 2005.

[2] J. E. Volder, “The CORDIC trigonometric computing technique,” IRE Trans. Electron. Computers, vol. 8, pp.330-334, Sept. 1959.

[3] N. Ide et al., “2.44-GFLOPS 300-MHz Floating-Point Vector-Processing Unit for High-Performance 3-D Computer Graphics Computing,” IEEE J. Solid-State Circuits, vol. 35, no. 7, pp. 1025-1033, July 2000.

[4] E. Lindholm et al., “A User-Programmable Vertex Engine,” Proc.ACM SIGGRAPH 2001, pp. 149-158, 2001.

[5] 조승현 외, "3D 그래픽스 가속 하드웨어 기술", 전자통신동향분석, 제22권, 제5호, 2007. 10.

[6] D.E. Eberly, 3D Game Engine Design. Morgan Kaufmann, 2000.

[7] U.J. Kapasi et al., “Programmable Stream Processors,” Computer, vol. 36, no. 8, pp. 54-62, Aug. 2003.

[8] W.J. Dally, P. Hanrahan, M. Erez, and T.J. Knight, “Merrimac: Supercomputing with Streams,” Proc. Supercomputing Conf., Nov. 2003.

[9] T. Moller and E. Haines, Real-Time Rendering, second ed. A.K. Peters Ltd., 2002.

[10] “Digital Arithmetic”, Morgan Kaufmann, 2003.

[11] “Elementary Functions: Algorithms and Implementations”, Birkhauser, 1997.

[12] C. S. Wu and A. Y. Wu, “Modified vector rotational CORDIC (MVR CORDIC)algorithm and architecture,” IEEE Trans. Circuits Syst. II,Analog Digit. Signal Process., vol. 48, no. 6, pp. 548 - 561, Jun. 2001.

[13] C. H. Lin, A. Y. Wu, “Mixed-scaling-rotation CORDIC(MSR-CORDIC) algorithm and architecture for highperformance vector rotational DSP applications,” IEEE Trans. Circuits and Systems Part-I: Fundamental Theory and Applications, Vol. 52, No. 11, pp. 2385 - 2396 , Nov. 2005.

[14] G.J. Hekstra and E.F. Deprettere, “Floating Point CORDIC,” Proc. 11th IEEE Symp. Computer Arithmetic, pp. 130-137, 1993.

[15] Y. H. Hu and S. Naganathan, “An angle recoding method for CORDIC algorithm implementation,” IEEE Trans. Computers, vol. 42, pp.99-102, Jan. 1993.

[16] Y. H. Hu, “CORDIC-based VLSI architectures for digital signal processing,”IEEE Signal Processing Mag., pp. 16 - 35, July 1992.