왜 3차원을 다루는가?

시리즈의 대부분은 2차원:

- 화면에서 보기 쉬움
- 직관적으로 이해하기 쉬움
- 핵심 아이디어를 명확하게 전달

하지만 가끔 3차원을 봐야 하는 이유:

- 모든 개념이 **고차원으로 자연스럽게 확장**됨
- 실제 응용 (컴퓨터 그래픽스, 로보틱스)은 3차원
- 2차원에서 배운 것이 일반적임을 확인

핵심 메시지: 2차원에서 이해하면 3차원(및 그 이상)은 쉽습니다!

3차원 선형변환의 시각화

2차원과 똑같은 원리:

- 입력: 3차원 벡터
- 출력: 3차원 벡터
- 모든 벡터가 동시에 움직임
- 공간 전체가 변형됨

시각화 방법:

1. **격자 사용:**
   - 3차원 공간에 격자선 그리기
   - xy평면, xz평면, yz평면의 격자
   - 변환 후 격자가 어떻게 변하는지 관찰
2. **기저벡터 추적:**
   - 원본 축(배경에 남겨둠)
   - 변환된 축 (어디로 가는지 추적)
   - 나머지는 따라옴

3차원 기저벡터

표준 기저:

```
î = [1, 0, 0]  (x축 방향)
ĵ = [0, 1, 0]  (y축 방향)
k̂ = [0, 0, 1]  (z축 방향)
```

모든 3D 벡터는 선형결합:

```
[x]   [1]   [0]   [0]
[y] = x[0] + y[1] + z[0]
```

```
[z]   [0]   [0]   [1]
```

3×3 행렬

2×2에서 3×3로:

2차원:

```
[a  b]
[c  d]
```

- 2개 열: î, ĵ의 변환
- 2개 행: 2차원 좌표

3차원:

```
[a  b  c]
[d  e  f]
[g  h  i]
```

- **3개 열**: î, ĵ, k̂의 변환
- **3개 행**: 3차원 좌표
- **9개 숫자**로 변환 완전 설명

의미:

- **첫 번째 열 [a, d, g]**: î가 어디로 가는가
- **두 번째 열 [b, e, f]**: ĵ가 어디로 가는가
- **세 번째 열 [c, f, i]**: k̂가 어디로 가는가

3차원 행렬-벡터 곱셈

공식:

```
[a  b  c] [x]   [ax + by + cz]
[d  e  f] [y] = [dx + ey + fz]
[g  h  i] [z]   [gx + hy + iz]
```

기하학적 의미:

```
결과 = x·(변환된 î) + y·(변환된 ĵ) + z·(변환된 k̂)
```

2차원과 완전히 똑같은 논리!

구체적 예제

예제 1: z축 중심 90도 회전

기저벡터의 변환:

- $\hat{\imath}$ = [1, 0, 0] → [0, 1, 0] (xy평면에서 90도 회전)
- $\hat{\jmath}$ = [0, 1, 0] → [-1, 0, 0]
- $\hat{k}$ = [0, 0, 1] → [0, 0, 1] (z축은 그대로)

```
행렬 = [0  -1  0]
       [1   0  0]
       [0   0  1]
```

의미:

- xy평면에서 회전
- z 높이는 유지
- 회전축 = $\hat{k}$

벡터 [1, 1, 1] 적용:

```
[0  -1  0] [1]   [0·1 + (-1)·1 + 0·1]   [-1]
[1   0  0] [1] = [1·1 + 0·1 + 0·1   ] = [1 ]
[0   0  1] [1]   [0·1 + 0·1 + 1·1   ]   [1 ]
```

결과: [1, 1, 1] → [-1, 1, 1]

3차원 행렬 곱셈

2차원과 같은 원리:

- M_2M_1 = "M_1 다음 M_2"
- 결과의 각 열 = 합성 후 기저벡터

계산 방법:

M_2M_1의 첫 번째 열 ($\hat{\imath}$의 최종 위치):

1. M_1의 첫 번째 열 = M_1 후 $\hat{\imath}$의 위치
2. 그것을 M_2로 변환
3. 결과가 M_2M_1의 첫 번째 열

같은 방식으로 두 번째, 세 번째 열도 계산

수치적으로:

```
(M₂M₁)ᵢⱼ = Σₖ (M₂)ᵢₖ (M₁)ₖⱼ
```

하지만 기하학적으로 이해하는 것이 더 좋습니다!

3차원 변환의 응용

1. 컴퓨터 그래픽스:

- 3D 객체 회전
- 카메라 시점 변경

- 애니메이션

2. 로보틱스:

- 로봇 팔의 움직임
- 좌표계 변환
- 센서 데이터 통합

3차원 회전이 중요한 이유:

- 복잡한 3D 회전을 설명하기 어려움
- 하지만 간단한 회전들의 **합성**으로 분해 가능
- 각 축 중심 회전을 행렬로 표현
- 행렬 곱셈으로 합성

예: 임의의 회전 = x축 회전 × y축 회전 × z축 회전

비정방 행렬 (Non-square Matrices)

질문: 3D → 2D 또는 2D → 3D 변환은?

가능합니다!

3D → 2D 변환

예: 3D 물체의 그림자

행렬 크기: **2×3** (2개 행, 3개 열)

```
[a  b  c]
[d  e  f]
```

의미:

- 입력: 3차원 벡터 [x, y, z]
- 출력: 2차원 벡터
- 3개 열: 3D 기저벡터가 2D로 어디로 가는가

예: xy평면으로 투영

```
[1  0  0]
[0  1  0]
```

벡터 [x, y, z] → [x, y] (z 좌표 버림)

기하학적 의미:

- 3차원 → 2차원으로 차원 축소
- 정보 손실 (z 좌표 사라짐)
- 예: 태양광(평행광선)에 의한 그림자

2D → 3D 변환

행렬 크기: **3×2** (3개 행, 2개 열)

```
[a  b]
[c  d]
[e  f]
```

의미:

- 입력: 2차원 벡터 [x, y]
- 출력: 3차원 벡터
- 2개 열: 2D 기저벡터가 3D로 어디로 가는가

예: xy평면에 2D를 임베딩

```
[1  0]
[0  1]
[0  0]
```

벡터 [x, y] → [x, y, 0]

기하학적 의미:

- 2차원을 3차원 공간의 평면에 배치
- 차원 확장 (하지만 3D 전체는 span 안 함)

행렬 크기 규칙

m×n 행렬:

- **n개 열**: 입력 차원 (n차원 벡터)
- **m개 행**: 출력 차원 (m차원 벡터)

예:

- 2×3 행렬: 3D → 2D
- 3×2 행렬: 2D → 3D
- 3×3 행렬: 3D → 3D
- 4×3 행렬: 3D → 4D

행렬 곱셈의 차원 호환성

M_1 × M_2가 가능하려면:

M_1이 m×n, M_2가 p×q일 때:

- **n = p**여야 함
- 결과는 **m×q** 행렬

이유:

- M_2의 출력(p차원) = M_1의 입력(n차원)이어야 함
- 첫 번째 변환의 출력을 두 번째 변환의 입력으로

예:

```
(2×3) × (3×4) = (2×4)  ✓
(3×2) × (3×3) = ???    ✗ (2 ≠ 3)
(3×3) × (3×2) = (3×2)  ✓
```

3차원에서의 선형성

2차원과 같은 정의:

1. **직선은 직선으로 유지**
   - 곡선이 되면 안 됨
   - 격자선이 평행하고 균등
2. **원점은 고정**
   - [0, 0, 0]은 [0, 0, 0]으로

결과:

- 평면은 평면으로 유지 (또는 선/점으로 축소)
- 평행한 평면들은 평행 유지

핵심 직관: 2D → 3D 확장

개념	2D	3D
기저벡터	î, ĵ (2개)	î, ĵ, k̂ (3개)
행렬 크기	2×2	3×3
좌표	[x, y]	[x, y, z]
행렬 원소	4개	9개
곱셈 공식	같은 논리	같은 논리
선형결합	av + bw	av + bw + cu

모든 것이 자연스럽게 확장됩니다!

n차원으로의 일반화

n차원 선형변환:

- n×n 행렬
- n개 기저벡터
- n^2 개 원소

하지만 시각화는 3D까지만:

- 4D 이상은 직접 볼 수 없음
- 하지만 수학적으로는 완전히 같은 방식

대수적 조작은 차원과 무관하게 작동:

- 행렬 곱셈
- 역행렬
- 행렬식
- 고유벡터
- 등등...

실전 팁

3D 변환 이해하기:

1. **한 번에 한 축씩:**
   - x축 회전만 먼저 이해
   - y축 회전만 이해
   - z축 회전만 이해
   - 합성으로 이해
2. **2D 단면 생각:**
   - xy평면에서 무슨 일이?
   - xz평면에서는?
   - yz평면에서는?
3. **기저벡터 추적:**
   - î, ĵ, k̂만 따라가기
   - 나머지는 자동

---

2. 퀴즈

문제 1

2차원 선형변환의 개념이 3차원으로 어떻게 확장되는지 설명하시오. 특히 행렬의 열이 나타내는 의미가 2D와 3D에서 어떻게 같은 논리를 따르는지 서술하시오.

문제 2

다음 3×3 행렬이 나타내는 선형변환을 기하학적으로 해석하시오:

```
[1  0  0]
[0  0 -1]
[0  1  0]
```

각 기저벡터 î, ĵ, k̂가 어디로 변환되는지 쓰고, 이 변환이 3차원 공간을 어떻게 변형시키는지 설명하시오.

문제 3

비정방 행렬(non-square matrix)에 대해 설명하시오. 2×3 행렬과 3×2 행렬의 차이를 기하학적 관점에서 서술하고, 각각의 예를 들어 어떤 종류의 변환을 나타내는지 설명하시오.

문제 4

두 행렬의 곱셈 AB가 정의되기 위한 조건을 설명하시오. 왜 A의 열 개수와 B의 행 개수가 같아야 하는지를 "변환의 합성" 관점에서 논리적으로 서술하시오.

문제 5

3차원 선형변환이 컴퓨터 그래픽스와 로보틱스에서 중요한 이유를 설명하시오. 특히 복잡한 3D 회전을 간단한 회전들의 합성으로 분해하는 것이 왜 유용한지 서술하시오.

---

3. 퀴즈 답

문제 1 답안

2D → 3D 확장의 핵심 원리:

2차원과 3차원의 선형변환은 **완전히 같은 논리**를 따릅니다.

2차원 (2×2 행렬):

```
M = [a  b]
    [c  d]
```

의미:

- **첫 번째 열 [a, c]**: 기저벡터 î = [1, 0]이 변환 후 어디로 가는가
- **두 번째 열 [b, d]**: 기저벡터 ĵ = [0, 1]이 변환 후 어디로 가는가

3차원 (3×3 행렬):

```
M = [a  b  c]
    [d  e  f]
    [g  h  i]
```

의미:

- **첫 번째 열 [a, d, g]**: 기저벡터 î = [1, 0, 0]이 변환 후 어디로 가는가
- **두 번째 열 [b, e, f]**: 기저벡터 ĵ = [0, 1, 0]이 변환 후 어디로 가는가
- **세 번째 열 [c, f, i]**: 기저벡터 k̂ = [0, 0, 1]이 변환 후 어디로 가는가

같은 논리의 확장:

1. **기저벡터의 개수:**
   - 2D: 2개 (î, ĵ)
   - 3D: 3개 (î, ĵ, k̂)
2. **행렬의 열 개수:**
   - 2D: 2개 열
   - 3D: 3개 열
   - 각 열 = 변환된 기저벡터
3. **좌표 개수:**

    - 2D: 2개 (x, y)
    - 3D: 3개 (x, y, z)
    - 각 좌표 = 해당 기저벡터의 스케일
4. **행렬-벡터 곱셈:**

2D:

```
[a  b] [x]   = x·[a] + y·[b]
[c  d] [y]       [c]     [d]
```

3D:

```
[a  b  c] [x]   = x·[a] + y·[b] + z·[c]
[d  e  f] [y]       [d]     [e]     [f]
[g  h  i] [z]       [g]     [h]     [i]
```

핵심 통찰: 변환을 완전히 설명하려면 **기저벡터가 어디로 가는지만** 알면 됩니다. 이것은 차원과 무관한 보편적 원리입니다.

일반화:

- n차원: n×n 행렬, n개 열, 각 열은 변환된 기저벡터
- 논리는 정확히 동일

이 일관성 덕분에 2차원에서 배운 모든 직관이 3차원(및 고차원)으로 자연스럽게 확장됩니다.

문제 2 답안

주어진 행렬:

```
[1  0  0]
[0  0 -1]
[0  1  0]
```

각 기저벡터의 변환:

î = [1, 0, 0]의 변환:

- 첫 번째 열 = [1, 0, 0]
- î → [1, 0, 0] (그대로)

ĵ = [0, 1, 0]의 변환:

- 두 번째 열 = [0, 0, 1]
- ĵ → [0, 0, 1] = k̂ (위쪽으로)

k̂ = [0, 0, 1]의 변환:

- 세 번째 열 = [0, -1, 0]
- k̂ → [0, -1, 0] = -ĵ (아래쪽으로)

기하학적 해석:

이 변환은 **x축 중심 90도 회전**입니다 (시계방향, 오른손 법칙 기준으로는 -90도).

시각화:

1. **x축 (î 방향):**
   - 변하지 않음
   - 회전축 역할
2. **yz평면에서:**
   - y축(ĵ) → z축(k̂)으로
   - z축(k̂) → -y축(-ĵ)으로
   - 90도 시계방향 회전 (x축을 바라볼 때)

전체 공간의 변형:

- x 좌표는 유지
- y와 z 좌표가 회전
- 오른손을 x축 방향으로 향하고 손가락을 y에서 z로 구부리면 90도 회전

예시 벡터:

벡터 [1, 1, 0] (xy평면):

```
[1  0  0] [1]    [1]
[0  0 -1] [1] = [0]
[0  1  0] [0]    [1]
```

→ [1, 0, 1] (xz평면으로 회전)

벡터 [0, 0, 1] (z축):

```
[1  0  0] [0]    [0]
[0  0 -1] [0] = [-1]
[0  1  0] [1]    [0]
```

→ [0, -1, 0] (-y축)

응용:

- 3D 그래픽스: 객체를 x축 중심으로 회전
- 로보틱스: 로봇 팔의 첫 번째 관절 회전

문제 3 답안

비정방 행렬(Non-square Matrix):

행렬의 행 개수와 열 개수가 다른 경우입니다. 이런 행렬은 **서로 다른 차원 사이의 변환**을 나타냅니다.

2×3 행렬 (3D → 2D):

```
A = [a  b  c]  (2개 행, 3개 열)
    [d  e  f]
```

의미:

- **입력:** 3차원 벡터 [x, y, z]
- **출력:** 2차원 벡터
- **3개 열:** 3D 기저벡터(î, ĵ, k̂)가 2D로 어디로 가는가

기하학적 해석:

- **차원 축소** (3D → 2D)
- 정보 손실 발생 (한 차원이 사라짐)
- 3D 공간을 2D 평면으로 "투영(projection)"

예시: xy평면으로 투영

```
[1  0  0]
[0  1  0]
```

작동:

```
[1  0  0] [x]    [x]
[0  1  0] [y] = [y]
          [z]
```

→ z 좌표를 버림

실제 예:

- 3D 객체의 그림자 (태양광이 평행할 때)
- 카메라의 2D 이미지 (3D 세계를 평면에 투영)
- 컴퓨터 화면 (3D 게임을 2D 모니터에 표시)

3×2 행렬 (2D → 3D):

```
B = [a  b]  (3개 행, 2개 열)
    [c  d]
    [e  f]
```

의미:

- **입력:** 2차원 벡터 [x, y]
- **출력:** 3차원 벡터
- **2개 열:** 2D 기저벡터(î, ĵ)가 3D로 어디로 가는가

기하학적 해석:

- **차원 확장** (2D → 3D)
- 하지만 3D 전체를 채우지는 못함
- 2D 평면을 3D 공간 안에 "임베딩(embedding)"

예시: xy평면에 배치

```
[1  0]
[0  1]
[0  0]
```

작동:

```
[1  0] [x]    [x]
[0  1] [y] = [y]
[0  0]        [0]
```

→ z = 0인 평면에 배치

실제 예:

- 2D 평면 그래픽을 3D 공간에 배치
- 2D 지도를 3D 지형에 매핑
- 평면 텍스처를 3D 모델에 적용

차이 요약:

특성	2×3 (3D→2D)	3×2 (2D→3D)
방향	차원 축소	차원 확장
정보	손실	보존
span	최대 2D	최대 2D 평면
역변환	불가능	불가능 (유일하지 않음)

문제 4 답안

행렬 곱셈 AB가 정의되기 위한 조건:

A가 **m×n** 행렬이고 B가 **p×q** 행렬일 때:

- **n = p**여야 함
- 결과는 **m×q** 행렬

조건의 의미:

- A의 **열 개수** = B의 **행 개수**

"변환의 합성" 관점에서 설명:

행렬 곱셈 AB는 "B를 먼저 적용하고 A를 나중에 적용"하는 합성 변환입니다.

단계별 분석:

1. **B의 역할:**
   - 입력: q차원 벡터
   - 출력: **p차원 벡터** (B의 행 개수)
2. **A의 역할:**
   - 입력: **n차원 벡터** (A의 열 개수)
   - 출력: m차원 벡터
3. **합성이 가능하려면:**
   - B의 출력 차원 = A의 입력 차원
   - 즉, **p = n**

시각적 설명:

```
q차원 벡터 --[B: p×q]--> p차원 벡터 --[A: m×n]--> m차원 벡터
                                                   (n = p)
```

B의 출력이 A의 입력으로 들어가야 하므로, 차원이 일치해야 합니다!

구체적 예시:

가능한 경우:

```
A: 2×3 (3D → 2D)
B: 3×4 (4D → 3D)

AB: (2×3) × (3×4) = 2×4
       ↑       ↑
      같음!
```

의미: 4D → 3D → 2D 변환

- B: 4차원 벡터를 3차원으로
- A: 3차원 벡터를 2차원으로
- AB: 직접 4차원에서 2차원으로

불가능한 경우:

```
A: 2×3 (3D → 2D)
B: 2×4 (4D → 2D)

AB: (2×3) × (2×4) = ???
          ↑    ↑
        3 ≠ 2
```

의미: B는 2차원 벡터를 출력하는데, A는 3차원 벡터를 입력으로 원함 → 연결 불가!

일반적 규칙:

합성 가능 여부:

```
(m×n) × (p×q) → n과 p 비교
```

- $n = p$: 가능, 결과는 $m \times q$
- $n \neq p$: 불가능

직관: 행렬 곱셈은 파이프 연결과 같습니다:

- 각 행렬 = 파이프의 한 부분
- 앞 파이프의 출구 크기 = 뒤 파이프의 입구 크기
- 크기가 맞지 않으면 연결 불가

이 조건은 변환의 합성이라는 기하학적 의미에서 자연스럽게 나오며, 단순한 규칙 암기가 아니라 **논리적 필연성**입니다.

문제 5 답안

3차원 선형변환이 중요한 이유:

1. 컴퓨터 그래픽스:

3D 객체 조작:

- 회전, 확대/축소, 반사
- 여러 객체의 상대적 위치
- 애니메이션 (시간에 따른 변환)

카메라와 시점:

- 카메라 위치 변경 = 좌표계 변환
- 시점 전환 = 역변환
- 3D → 2D 투영 (화면에 표시)

예: 게임 캐릭터 회전

- 캐릭터의 모든 정점(vertex)에 회전 행렬 적용
- 실시간으로 수천/수만 번 계산
- 행렬 연산이 GPU에 최적화됨

2. 로보틱스:

로봇 팔(Manipulator):

- 각 관절의 회전 = 선형변환
- 전체 움직임 = 변환의 합성
- 말단(end-effector)의 위치 계산

좌표계 변환:

- 센서 좌표계 → 로봇 좌표계
- 로봇 좌표계 → 세계 좌표계
- 여러 센서 데이터 통합 (센서 퓨전)

예: SLAM (Simultaneous Localization and Mapping)

- 로봇의 자세(pose) = 3D 변환
- 센서 측정을 세계 좌표로 변환
- 지속적인 좌표계 업데이트

3. 복잡한 회전 분해의 중요성:

문제: 임의의 3D 회전은 직관적이지 않음

- 어떤 축 중심?
- 얼마나 회전?
- 시각화하기 어려움

해결: 간단한 회전들의 합성

오일러 각(Euler Angles): 임의의 회전 = x축 회전 × y축 회전 × z축 회전

```
R_총 = R_z(ψ) × R_y(θ) × R_x(φ)
```

각 회전은 이해하기 쉬움:

- R_x: x축 중심 회전 (pitch)
- R_y: y축 중심 회전 (yaw)
- R_z: z축 중심 회전 (roll)

장점:

1. **이해 용이:**
    - 각 단계가 직관적
    - "먼저 x로 돌리고, 그 다음 y로..."
    - 복잡한 것을 단순하게 분해
2. **계산 효율:**
    - 각 축 회전 행렬은 간단
    - 미리 계산해두고 재사용
    - 행렬 곱셈으로 합성
3. **제어 가능:**
    - 각 회전 각도를 독립적으로 조절
    - 애니메이션 키프레임 설정
    - 로봇 관절 각도 제어
4. **일반성:**
    - 모든 3D 회전을 표현 가능
    - 수학적으로 증명됨
    - 범용적 방법

실제 응용 예:

항공기 자세:

```
Roll (φ): 날개 좌우 기울임
Pitch (θ): 기수 상하 움직임
Yaw (ψ): 좌우 방향 전환
```

```
자세 행렬 = R_z(ψ) × R_y(θ) × R_x(φ)
```

로봇 팔:

```
관절1 회전 → M₁
관절2 회전 → M₂
관절3 회전 → M₃

말단 위치 = M₃ × M₂ × M₁ × [초기위치]
```

결론:

3차원 선형변환은 현대 기술의 핵심입니다:

- 시각화가 가능한 최고 차원
- 실제 물리 세계와 직접 대응
- 복잡한 변환을 간단한 것들로 분해 가능
- 행렬 연산으로 효율적 계산

특히 "복잡한 것을 간단한 것의 합성으로"라는 아이디어는 선형대수의 핵심 철학이며, 3D 회전이 완벽한 예시입니다.