과제5 - 3D Illusion!

22011831 김형규

풀이

기본적인 아이디어는, 원본 이미지의 네 꼭짓점을 큐브의 각 네 개의 꼭짓점으로 보낼 수 있는 행렬을 찾아야 한다. 원본 이미지의 좌표를 (x, y), 큐브의 꼭짓점의 좌표를 (x`, y`)이라고 하면,

$$\begin{bmatrix} w` \cdot x` \\ w` \cdot y` \\ w` \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

원본 이미지의 네 좌표에 대하여 위의 식을 모두 만족하는 행렬 H를 찾는 것이 목적이다. H_{33}이 1이라고 가정하면, 미지수는 총 8개, 한 좌표쌍마다 방정식이 두 개씩 나오므로, 총 8개의 방정식이다. 미지수와 방정식의 개수가 같으므로, 행렬 H를 정확히 찾을 수 있다. 또한, 위의 식을 풀어 쓴 뒤, 행렬 연산으로 표현하면 다음과 같다.

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1 \cdot x`_1 & -y_1 \cdot x`_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1 \cdot y`_1 & -y_1 \cdot y`_1 \end{bmatrix} \begin{bmatrix} H_{11} \\ H_{12} \\ H_{13} \\ H_{21} \\ H_{22} \\ H_{23} \\ H_{31} \\ H_{32} \end{bmatrix} = \begin{bmatrix} x`_1 \\ y`_1 \end{bmatrix}$$

좌변의 왼쪽 행렬을 A, 가운데 행렬을 h, 우변의 행렬을 B라고 하면 다음 식이 성립한다.

$$Ah = B$$

이 식의 좌변과 우변의 왼쪽에 A의 역행렬을 곱해주면 좌변은 h만 남고, 우변은 $A^{-1} \cdot B$

가 된다. 즉, 원본 이미지의 네 꼭짓점을 큐브의 네 꼭짓점으로 보내는 행렬을 $A^{-1} \cdot B$

이다.

코드로 구현하기

먼저, A행렬과 B행렬을 초기화 해준다. 행렬 초기화는 반복문을 이용하면 어렵지 않게 구현할 수 있다. A행렬과 B행렬은 각각 8x8, 8x1크기의 행렬이다.

```
//A행렬의 짝수번호 행 초기화
A[2 * i][0] = p.x;
A[2 * i][1] = p.y;
A[2 * i][2] = 1;
A[2 * i][3] = 0;
A[2 * i][4] = 0;
A[2 * i][5] = 0;
A[2 * i][6] = -pts.pos[i].x * p.x;
A[2 * i][7] = -pts.pos[i].x * p.y;

//A행렬의 홀수번호 행 초기화
A[2 * i + 1][0] = 0;
A[2 * i + 1][1] = 0;
A[2 * i + 1][2] = 0;
A[2 * i + 1][3] = p.x;
A[2 * i + 1][4] = p.y;
A[2 * i + 1][5] = 1;
A[2 * i + 1][6] = -pts.pos[i].y * p.x;
A[2 * i + 1][7] = -pts.pos[i].y * p.y;
```

```
//B행렬값 초기화
for (int i = 0; i < 4; i++) {
    B[2*i][0] = pts.pos[i].x;
    B[2*i+1][0] = pts.pos[i].y;
}
```

왼쪽 이미지는 행렬 A를 초기화 하는 행렬이다. 반복문을 이용하여 해당하는 꼭짓점에 맞게 초기화를 해주었다. B행렬도 마찬가지 방법으로 초기화 해준다.

그 다음, A행렬의 역행렬을 구하여 invA에 저장해준다. invA는 미리 구현되어 있는 8x8행렬의 역행렬을 구하는 함수를 이용하였다.

```
//A의 역행렬을 구해서 invA에 초기화
InverseMatrixGJ8(A, invA);
```

이 함수를 이용하면 8x8크기의 A행렬의 역행렬을 invA에 저장할 수 있다.

그 다음, invA행렬과 B행렬을 곱해주어야 한다. invA행렬의 열 개수와 B행렬의 행 개수가 같으므로 곱셈을 진행할 수 있다.

```
//h = invA * B이다.
for (int i = 0; i < 8; i++) {
    h[i][0] = 0.0f;
    for (int j = 0; j < 8; j++) {
        h[i][0] += invA[i][j] * B[j][0];
    }
}
```

그리고 이 연산으로 생기는 행렬은 우리가 구하고자 하는 h행렬이므로, h에 저장해준다.

구한 h행렬을 토대로 큐브에 그림을 그려주면 된다. 먼저, 가독성을 높이기 위해 8x1크기인 h행렬을 3x3크기로 바꿔준다. h행렬의 첫 번째 원소부터 차례대로 넣어주면 되는데, 3x3행렬의 마지막 위치의 원소는 1이 된다.

```
//h행렬을 3by3행렬로 바꾼 행렬 M
float M[3][3];
for (int i = 0; i < 8; i++) {
    M[i/3][i%3] = h[i][0];
}
//M의 마지막 원소는 1이다.
M[2][2] = 1;
```

이제 행렬 M을 토대로 각 좌표를 변환해서 그려주면 된다.

시행착오

원래 주어진 행렬A는 8x9행렬이고, h는 9x1행렬이다. 최종목적은 두 행렬의 곱이 0이 되는 행렬 h를 찾는 것이다. A는 행 개수와 열 개수가 다르기 때문에 정상적인 방법으로 역행렬을 구할 수는 없고, 역행렬과 비슷한 역할을 하는 "의사 역행렬"을 구해야 한다. 의사 역행렬을 구하려면, 행렬 A를 특이값 분해(Singular Value Decomposition)해야 한다. 특이값 분해란, 행렬 A를 U, S, V행렬 3개로 분해하는 것인데, V행렬의 마지막 열벡터가 최종 구하고자 하는 h행렬이 된다. A의 역행렬을 구하는 것 자체는 충분히 할 수 있지만, 특이값 분해 과정은 너무나 어려워서 할 수 없었다.

참고자료

<Homography Estimation> - CSE 252A, Winter2007, David Kriegman

Homography Matrix 추정 - https://ichi.pro/ko/homography-matrix-chujeong-90350826572588

공돌이의 수학 정리노트(의사역행렬의 기하학적 의미)

- https://angeloyeo.github.io/2020/11/11/pseudo_inverse.html

다크 프로그래머(특이값 분해의 활용) - https://darkpgmr.tistory.com/106