이중선형 보간법을 이용한 이미지 회전

이름 김준호
국민대학교 전자공학부
jhk00@kookmin.ac.kr

요 약

본 보고서는 이미지 회전을 위해 이중선형 보간법을 활용하는 방법에 대해 기술하였다. 이중선형 보간법은 회전된 이미지의 픽셀 값을 부드럽게 보정하여 원본 이미지와 회전된 이미지 간의 차이를 최소화하는 방법이다. 수행한 과제에서는 입력 이미지와 회전 각도를 기반으로 회전 행렬을 계산하고, 회전된 이미지의 크기와 좌표 오프셋을 계산하여 출력 이미지를 생성하였다. 이후 이중선형 보간법을 적용하여 출력 이미지의 픽셀 값을 결정하였다. 실험 결과, 이중선형 보간법을 사용한 이미지 회전은 고해상도 및 자연스러운 회전 결과를 제공함을 확인하였다.

1. 서론

이미지 회전은 컴퓨터 비전 및 컴퓨터 그래픽스 분야에서 널리 사용되는 중요한 기술 중 하나이다. 이미지 회전은 이미지의 방향을 변경하여 시각적인 효과를 부여하거나 이미지 분석 및 처리에 활용할 수 있다. 그러나 이미지 회전은 픽셀 값의 이동과 보정 과정에서 정보 손실이 발생할 수 있다는 문제가 있다.

이미지 보간은 디지털 이미지 처리에서 중요한 주제 중 하나로, 이미지의 해상도 조정, 회전, 변형, 복원 등 다양한 응용 분야에서 활용되고 있습니다. 이미지 보간은 이미지의 공간적인 연속성을 유지하면서 픽셀 값의 빠진 부분을 채워줌으로써 원래의 이미지와 비슷한 품질의 출력 이미지를 생성하는 방법이다.

최근접 이웃 보간법은 보간하려는 픽셀 위치에 가장 가까운 픽셀 값을 사용하여 값을 대체한다. 이 방법은 구현이 간단하고 계산 효율적이지만, 보간된 이미지의 부자연스러움과 픽셀 값의 계단 현상을 초래할 수 있다.

선형 보간법은 픽셀 값 사이를 직선으로 연결하여 보간하는 방법이다. 이 방법은 최근접 이웃 보간법보다 더 부드러운 결과를 제공하고, 선형적인 변화를 갖는 이미지를 생성한다.

이중선형 보간법은 선형 보간법을 확장한 방법으로, 픽셀 값을 주변 4 개의 픽셀로부터 선형적으로 보간한다. 이 방법은 주변 픽셀의 가중 평균을 사용하여 더 정확한 보간을 수행하며, 엣지 부근에서도 자연스러운 결과를 얻을 수 있다.

이차선형 보간법은 픽셀 값 사이를 이차원적으로 보간하여보다 부드러운 결과를 얻을 수 있다.

본 보고서에서는 이중선형 보간법을 적용하여 이미지 회전과 보간 과정에서의 방법을 보인다. 이중선형 방법을 적용하는 방법말고 다른 방법으로는 이차선형 보간법을 사용하는 방법이 있다. 그러나 이차선형 보간법은 계산량이 많고 엣지 부근에서 부정확한 결과를 낼 수 있다는 한계가 있기 때문에 이중선형 보간법을 이용해도 충분히 좋은 결과를 낼 수 있다.

본 보고서에서는 입력 이미지와 회전 각도를 기반으로 회전 행렬을 계산하고, 회전된 이미지의 크기와 좌표 오프셋을 결정한다. 이후 이중선형 보간법을 적용하여 출력 이미지의 픽셀 값을 보정한다. 실험 결과, 이중선형 보간법은 고해상도 및 자연스러운 회전 결과를 제공함을 확인하였다.

2. 과제 수행 내용

2.1 회전행렬 계산(60 도)

좌표 (x, y)에 회전행렬을 곱하면 좌표평면상에서 Θ 만큼 회전한 좌표인 (x', y')를 구할 수 있다.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$ 수식 1.

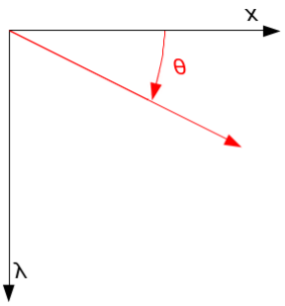

그림 1.

이미지를 행렬로 표현하면 x 축은 열이 되고 y 축은 행이 된다. 그래서 이해를 돕기 위해 위의 그림 2 를 돌려놨다. 이번 과제는 이미지를 60 도로 회전하는 것이기 때문에 Θ에 60 도를 대입하고 회전행렬을 계산한다.

2.2 회전된 이미지의 좌표 계산

원본 이미지의 가로세로크기를 구하고 원본 이미지의 꼭지점 픽셀들을 구한다. 이때 원본 이미지의 크기는 세로 1124, 가로 843 이 된다. 원본 이미지의 꼭짓점들은 [0,0], [0,1123], [842, 0], [842, 1123]이 된다.

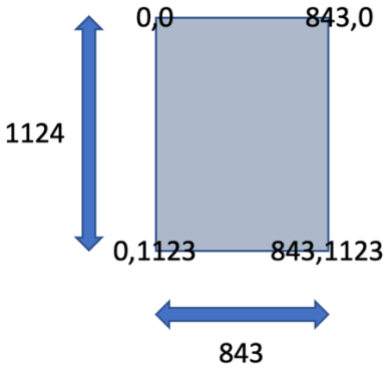

그림 2.

원본 이미지의 꼭짓점 좌표들로 행렬을 만들면 수식 2 가 된다.

$$\begin{bmatrix} 0 & 0 \\ 0 & 1123 \\ 843 & 0 \\ 843 & 1123 \end{bmatrix}$$

수식 2.

회전된 이미지의 각 꼭짓점 좌표들은 수식 1 에 원본 이미지의 좌표들을 대입해 구한다. 이를 통해 계산된 좌표들은 [0, 0], [-972.54651099, 561.5], [421, 729.1933769], [-551.54651099 1290.6933769]이다. 아래 그림 3 에서는 편의를 위해 소수점을 버리고 표시했다.

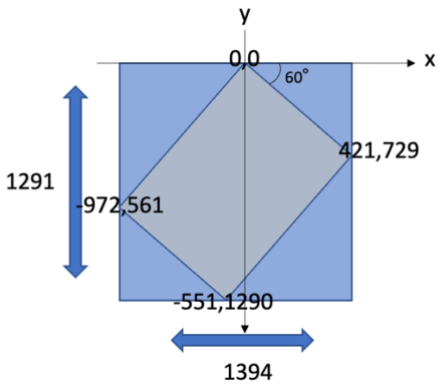

그림 3.

회전된 이미지의 좌표들을 아래 수식 3 과 같은 행렬로 표현 할 수 있다.

$$\begin{bmatrix} 0 & 0 \\ -972.54651099 & 561.5 \\ 421 & 729.1933769 \\ -551.54651099 & 1290.6933769 \end{bmatrix}$$

수식 3.

2.3 회전된 이미지의 크기와 오프셋 계산

회전된 이미지의 꼭짓점들인 수식 3 은 음의 값을 가지면 안되므로 각 좌표에 오프셋들을 더해줘서 음의 값을 가지지 않게 한다. 수식 3 에서 행을 기준으로 첫번째 행과 두번째 행의 최소, 최대값들을 구하고 정수형으로 형 변환한다. 이때 좌표들의 소수점이 버려지면서 데이터의 손실이 발생한다.

수식 3 에 오프셋을 더해 회전된 이미지에 음의 좌표가 없게 할 수 있다. 이때 오프셋은 행의 최소값들의 부호를 바꾸면 구할 수 있다. 오프셋은 가로 세로 (972, 0)이 된다.

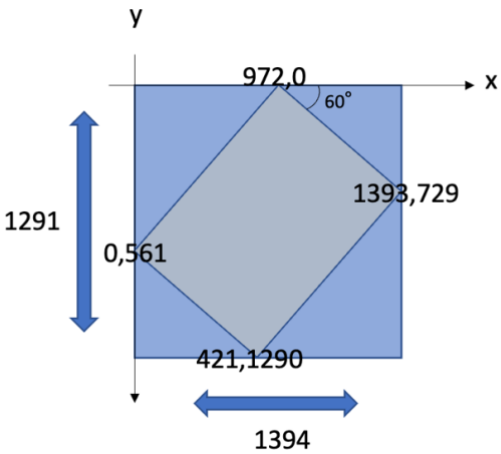

그림 4.

위의 그림 4 는 그림 3 에 오프셋을 더한 그림이다. 이를 통해 오프셋을 더해줘서 음의 좌표들을 없애 줌으로서 이미지가 잘리는 부분이 없도록 한다.

회전된 이미지의 크기는 행을 기준으로 구한 최대, 최솟값을 이용해 최대값 – 최솟값 + 1 을 하면 구할 수 있다. 이때 첫번째 행을 이용해 구한 값은 회전된 이미지의 가로 크기가 되고 두번째 행을 이용해 계산해준 값은 세로 크기가 된다. 회전된 이미지의 크기는 세로 1291, 가로 1394 가 된다.

2.4 이중선형 보간법을 이용해 회전된 이미지 계산

$$\begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix} \times \begin{bmatrix}x \\ y\end{bmatrix} + \begin{bmatrix}offsetx \\ offsety\end{bmatrix} = \begin{bmatrix}x' \\ y'\end{bmatrix}$$ 수식 4.

$$\begin{bmatrix}\cos\theta & \sin\theta \\ -\sin\theta & \cos\theta\end{bmatrix} \times \left(\begin{bmatrix}x' \\ y'\end{bmatrix} - \begin{bmatrix}offsetx \\ offsety\end{bmatrix}\right) = \begin{bmatrix}x \\ y\end{bmatrix}$$ 수식 5.

$$([x' \quad y'] - [offsetx \quad offsety]) \times \begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix} = [x \quad y]$$ 수식 6.

수식 4 를 보면 원본 이미지의 좌표 x, y 에 회전행렬을 곱하고 오프셋을 더해줘서 회전된 이미지의 좌표 x', y'을 구할 수 있다. 이때 오프셋은 [offsetx offsety]이고 값은 [972 0]이 된다. 수식 4 에 오프셋 벡터를 양변에 빼주고 회전행렬의 역행렬을 양변에 곱해주면 수식 5 를 구할 수 있다. 수식 6 은 수식 5 의 x, y 를 벡터 형태인 2x1 행렬에서 1x2 로 바꿔서 표현한 식으로 수식 5 와 같다.

회전된 이미지의 크기안에 들어있는 모든 픽셀들을 수식 6 의 x',y'에 대입해준다. 이후 얻은 원본 이미지의 좌표 x, y 가 음의 값을 가지거나 원본 이미지의 크기를 벗어나면 않는다는 조건을 만족하는 원본 이미지의 좌표들만을 얻는다.

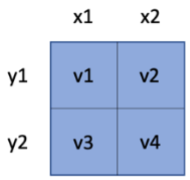

그림 5.

$$x_1 = \lfloor x \rfloor, x_2 = x_1 + 1, y_1 = \lfloor y \rfloor, y_2 = y_1 + 1$$ 수식 4.

$$\alpha = x - x_1, \beta = y - y_1$$ 수식 5.

$$v = (1-\beta)\{(1-\alpha)v_1 + \alpha v_2\} + \beta\{(1-\alpha)v_3 + \alpha v_4\}$$ 수식 6.

앞의 과정을 통해 얻은 원본 이미지의 좌표값을 x, y 라고 하면 x, y 의 소수점 자리를 내린 x1, y1 을 먼저 구한다. 그리고 x1, y1 에 각각 1 씩 더한 x2, y2 를 얻는다. 이상이 수식(4)에 대한 설명이다. 이때 x, y 는 그림 5 의 좌표안에 존재할 것이다. 수식 5 는 좌표 x, y 가 얼만큼 치우쳐졌는지 알기 위해 비율 α, β 를 계산한 것이다. 그리고 이중선형 보간법인 수식 6 에 구한값들을 대입하여 원본이미지에서 픽셀값을 선형근사하여 가져온다. 이때 그림 5 에 있는 v1, v2, v3, v4 는 원본이미지의 각 픽셀 값(B,G,R)이다. 파이썬 넘파이는 브로드캐스팅을 지원하므로 각 BGR 값들을 같이 계산해준다. 이렇게 원본이미지에서 선형근사된 픽셀값을 회전된 이미지의 좌표값인 x', y'의 픽셀값에 대입 해준다.

위의 과정을 그림 4 의 모든 픽셀들에 반복해주면 회전된 이미지를 얻을 수 있다.

3. 실험 결과 및 분석

그림 6.

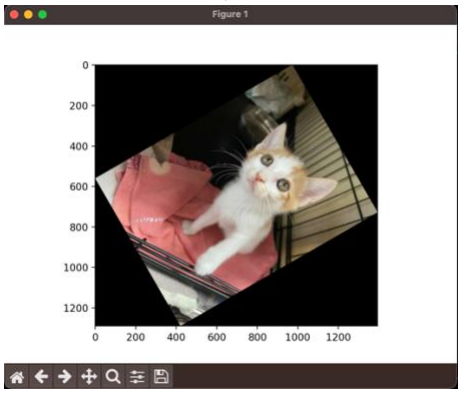

그림 7.

원본 이미지는 그림 6 이고 원본 이미지를 60 도 회전시켜 얻은 이미지는 그림 7 이다. 그림 7 을 보면 이중선형 보간법을 이용해 얻은 이미지가 눈으로 보기에 원본 이미지와 같은 것처럼 보인다. 앞에 회전된 이미지의 좌표를 구하고 정수형으로 형변환하는 과정에서 소수점을 버리면서 이미지의 정보가 손실되는 과정이 있었지만, 이중선형 보간법을 이용해 원본 이미지의 픽셀값을 선형근사해 회전된 이미지의 픽셀값에 대입하는 과정을 통해 원본이미지와 매우 비슷한 회전된 이미지를 구할 수 있었다.

4. 결론

회전행렬(60 도) 계산, 회전된 이미지의 좌표계산, 회전된 이미지의 크기와 오프셋 계산, 이를 통해 이

중선형 보간법을 이용해 회전된 이미지를 구하면서 원본 이미지를 회전시킨 이미지를 얻을 수 있었다. 사람의 눈으로 보기에는 원본 영상과 회전된 영상이 거의 차이가 없기에 이중선형 보간법으로도 충분히 크기가 같고 회전만된 이미지를 잘 보간할 수 있다는 것을 알 수 있다. 본 과제 보고서에서 사용된 수식(1)과 그림 1 빼고 전부 직접 본 저자가 파워포인트와 http://www.hostmath.com 를 통해 만들었음을 밝힌다.

참고문헌

[1] 다크프로그래머 블로그, “선형 보간법 (linear, bilinear, trilinear interpolation)”2014.01.14 게시, 2023.05.09 조회, https://darkpgmr.tistory.com/117

[2] betterdream 블로그, linear bilinear interpolation 개념정리, 2017.03.01 게시, 2023.05.09 조회, https://blog.naver.com/aorigin/220947541918

[3] 전파거북이 블로그, 회전 행렬(Rotation Matrix), 2020.08.09 게시, 2023.05.09 조회, https://ghebook.blogspot.com/2020/08/blog-post.html