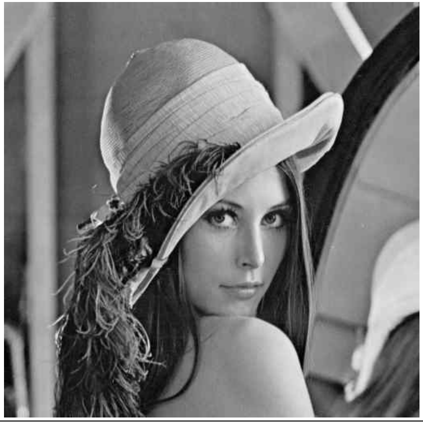
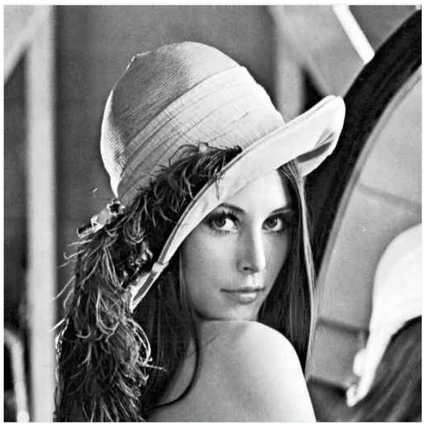
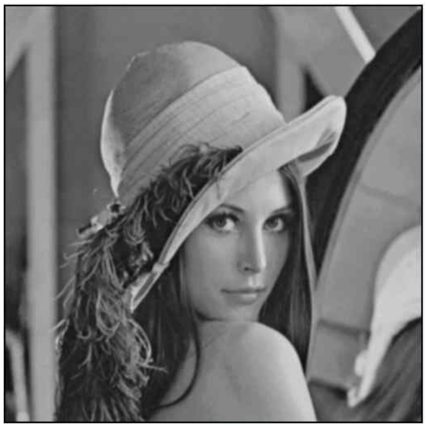
Computer Graphics

Assignment 4.

컴퓨터과학과
2013147513 조영재

1. Image Processing

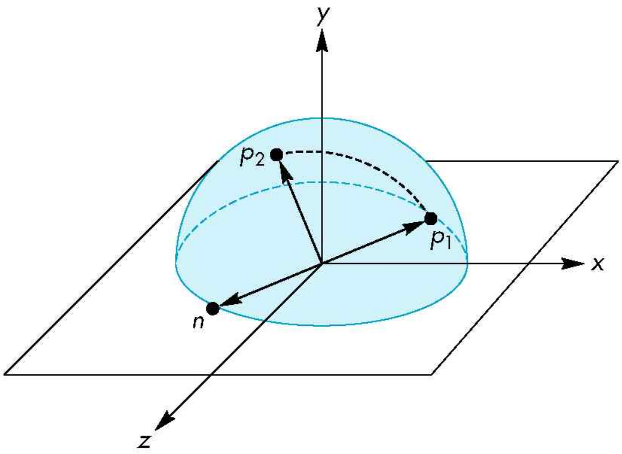
필터	방법	결과
Contrast Stretching	After obtaining the histogram of the image, make a linear function so that the grayscale changes from 0 to 255, bounding the minimum and maximum values. It then converts to grayscale corresponding to grayascale of original image.	
Histogram Equalization	Obtain the grayscale histogram of the image, and then obtain the cumulative probability distribution. This cumulative probability distribution becomes a kind of transform function, and the result is multiplied by 255 to obtain the actual grayscale value.	
Gaussian Filter	The Gaussian filter is a filter having a Gaussian distribution with a higher weight as the pixel is closer. The results of using the 5x5 Gaussian filter in the lecture material are shown on the right. When calculating with a filter, the edges are filled with black pixels because the edges need pixels outside the image. The result has a black border.	

3x3 Median Filter	Use a 3x3 median filter to sort 3x3 pixels around the pixel and replace it with a pixel with an intermediate grayscale value. Like the Gaussian filter, the edge does not calculate the value, so you can see the black border.	
7x7 Median Filter	It is the same as the 3x3 median filter, but the filter size is only 7x7.	

2. Quaternion Rotation

To create a virtual trackball, we need to translate the mouse movements of the plane into space movements. To realize this, I think there is a hemisphere placed on the xy plane. The mouse movement of the plane is regarded as a moving point of the x, y plane, and projecting it as a point on the hemisphere is the motion of the mouse in space

If the point on the plane of the mouse is outside the circle, the closest x, y coordinates of the circle at that point and the point where the z coordinate is set to 0 are considered to be mouse points in space.

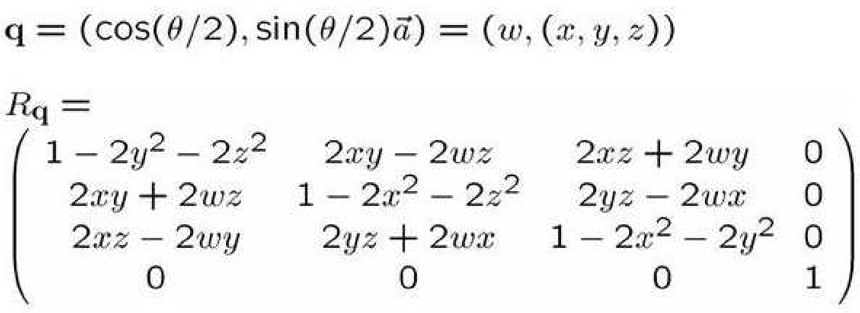


자료출처: http://vip.cs.utsa.edu/classes/cs5113s2007/lectures/materials/TrackBallRotation.html

Therefore, if you know the starting and ending points of a mouse, you can think of two vectors that have two points on the surface of the sphere, with each point as the end point and the origin as the starting point. In this case, the object can be rotated by rotating the outer product of the two vectors as the rotation axis and the angle of incidence as the rotation angle.

Let u and v be the two vectors, and the angle theta = arcsin (| n | / (| u || v |)) when the normal vector is n.

The rotation transformation matrix for this rotation is as follows.



$$\mathrm{q} = (\cos(\theta/2), \sin(\theta/2)\vec{a}) = (w, (x, y, z))$$

$$R_{\mathrm{q}} = \begin{pmatrix} 1-2y^2-2z^2 & 2xy-2wz & 2xz+2wy & 0 \\ 2xy+2wz & 1-2x^2-2z^2 & 2yz-2wx & 0 \\ 2xz-2wy & 2yz+2wx & 1-2x^2-2y^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

자료출처 :http://groups.csail.mit.edu/graphics/classes/6.837/F01/Lecture09/Slide24.html

▶ Implementation process

The main matrixes are as follows.
NewMatrix: the rotation transformation matrix computed after the mouse moves
OldMatrix: the rotation transformation matrix before the mouse moves
Quaternion: oldMatrix * newMatrix

As soon as the mouse is clicked on the **mouse** function, the previous quaternion matrix is copied and ready to be multiplied with a new rotation transformation matrix obtained when the mouse is moved. In addition, whether the point of the mouse is a point outside the circle or not corresponds to the point in space.

After that, the movement of the mouse is tracked in the **mouseMotion** function to correspond to a point on the space, and a rotation transformation matrix is obtained through a series of processes described above along with a normal vector and a rotation angle. The obtained matrix is multiplied by oldMatrix to obtain a quaternion matrix, which is a new rotation matrix.

The quaternion matrix thus obtained is applied to the real model through glMultMatrix in the **display** function.