JAVA DIP - IMAGE SHAPE CONVERSION

The shape of the image can easily be changed by using OpenCV. Image can either be flipped, scaled, or rotated in any of the four directions.

In order to change the shape of the image, we read the image and convert into Mat object. Its syntax is given below:

```
File input = new File("digital_image_processing.jpg");
BufferedImage image = ImageIO.read(input);
//convert Buffered Image to Mat.
```

Flipping an Image

OpenCV allows three types of flip codes which are described below:

Sr.No.	Flip Code
1	**0** 0 means, flipping around x axis.
2	**1** 1 means, flipping around y axis.
3	**-1** -1 means, flipping around both axis.

We pass the appropriate flip code into method **flip** in the **Core** class. Its syntax is given below:

```
Core.flip(source mat, destination mat1, flip_code);
```

The method **flip** takes three parameters: the source image matrix, the destination image matrix, and the flip code.

Apart from the flip method, there are other methods provided by the Core class. They are described briefly:

Sr.No.	Methods
1	**add***Matsrc1, Matsrc2, Matdst* It calculates the per-element sum of two arrays or an array and a scalar.
2	**bitwise_and***Matsrc1, Matsrc2, Matdst* It calculates the per-element bit-wise conjunction of two arrays or an array and a scalar.

3

bitwise_not*Matsrc, Matdst*

It inverts every bit of an array.

4

circle*Matimg, Pointcenter, intradius, Scalarcolor*

It draws a circle.

5

sumElems*Matsrc*

It blurs an image using a Gaussian filter.

6

subtract*Matsrc1, Scalarsrc2, Matdst, Matmask*

It calculates the per-element difference between two arrays or array and a scalar.

Example

The following example demonstrates the use of Core class to flip an image:

```
import java.awt.image.BufferedImage;
import java.awt.image.DataBufferByte;

import java.io.File;
import javax.imageio.ImageIO;

import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;

import org.opencv.imgproc.Imgproc;

public class Main {
   public static void main( String[] args ) {

      try {
         System.loadLibrary( Core.NATIVE_LIBRARY_NAME );
         File input = new File("digital_image_processing.jpg");
         BufferedImage image = ImageIO.read(input);

         byte[] data = ((DataBufferByte) image.getRaster().  getDataBuffer()).getData();
         Mat mat = new Mat(image.getHeight(),image.getWidth(),CvType.CV_8UC3);
         mat.put(0, 0, data);

         Mat mat1 = new Mat(image.getHeight(),image.getWidth(),CvType.CV_8UC3);
         Core.flip(mat, mat1, -1);

         byte[] data1 = new byte[mat1.rows()*mat1.cols()*(int)(mat1.elemSize())];
         mat1.get(0, 0, data1);
         BufferedImage image1 = new BufferedImage(mat1.cols(), mat1.rows(), 5);
         image1.getRaster().setDataElements(0,0,mat1.cols(),mat1.rows(),data1);

         File ouptut = new File("hsv.jpg");
         ImageIO.write(image1, "jpg", ouptut);

      } catch (Exception e) {
         System.out.println("Error: " + e.getMessage());
      }
   }
```

```
}
```

Output

When you run the above example, it would flip an image name **digital_image_processing.jpg** to its equivalent HSV color space image and write it on hard disk with name **flip.jpg**.

Original Image

Flipped Image