IMAGE PROCESSING PROGRAMS (cv2 + PIL)

1) Loading Image (cv2)

```
import cv2
img = cv2.imread("image.jpg")
cv2.imshow("Loaded Image", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Loading Image (PIL)

```
from PIL import Image
img = Image.open("image.jpg")
img.show()
```

2) Gray Level Conversion / Format Change

```
import cv2
img = cv2.imread("image.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imwrite("gray.jpg", gray)
```

```
from PIL import Image
img = Image.open("image.jpg")
img.save("image.png")
```

3) Cropping Image

```
import cv2
img = cv2.imread("image.jpg")
crop = img[50:200, 50:200]
cv2.imwrite("crop.jpg", crop)
```

```
from PIL import Image
img = Image.open("image.jpg")
crop = img.crop((50,50,200,200))
crop.save("crop.jpg")
```

4) Geometric Transformations (Resize, Rotate)

```
import cv2
img = cv2.imread("image.jpg")
res = cv2.resize(img, (300,300))
cv2.imwrite("resize.jpg", res)
```

```
rot = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
cv2.imwrite("rotate.jpg", rot)
```

5) Thresholding

```
import cv2
img = cv2.imread("image.jpg",0)
_,th = cv2.threshold(img,120,255,cv2.THRESH_BINARY)
cv2.imwrite("threshold.jpg", th)
```

6) Negative Transformation

```
import cv2
img = cv2.imread("image.jpg")
neg = 255 - img
cv2.imwrite("negative.jpg", neg)
```

7) Log Transformation

```
import cv2, numpy as np
img = cv2.imread("image.jpg",0)
log_img = np.log1p(img)
log_img = (255*log_img/log_img.max()).astype('uint8')
cv2.imwrite("log.jpg", log_img)
```

8) Gamma Transformation

```
import cv2, numpy as np
img = cv2.imread("image.jpg",0)
gamma = 2.2
```

```
gamma_img = ((img/255)**gamma)*255
gamma_img = gamma_img.astype('uint8')
cv2.imwrite("gamma.jpg", gamma_img)
```

9) Sobel Edge Detection

```
import cv2
img = cv2.imread("image.jpg",0)
sobelx = cv2.Sobel(img,cv2.CV_64F,1,0)
sobely = cv2.Sobel(img,cv2.CV_64F,0,1)
sobel = cv2.magnitude(sobelx,sobely)
cv2.imwrite("sobel.jpg", sobel)
```

10) Canny Edge Detection

```
import cv2
img = cv2.imread("image.jpg",0)
edges = cv2.Canny(img,100,200)
cv2.imwrite("canny.jpg", edges)
```

11) Prewitt Edge Detection

```
import cv2, numpy as np
img = cv2.imread("image.jpg",0)
kx = np.array([[1,0,-1],[1,0,-1],[1,0,-1]])
ky = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])
px = cv2.filter2D(img,-1,kx)
py = cv2.filter2D(img,-1,ky)
prewitt = px + py
cv2.imwrite("prewitt.jpg", prewitt)
```

12) Histogram Equalization

```
import cv2
img = cv2.imread("image.jpg",0)
eq = cv2.equalizeHist(img)
cv2.imwrite("equalized.jpg", eq)
```

13) CLSF (Constrained Least Square Filtering)

```
import cv2, numpy as np
def clsf(img, psf, gamma):
    G = np.fft.fft2(img)
    H = np.fft.fft2(psf, img.shape)
    P = np.fft.fft2([[0,-1,0],[-1,4,-1],[0,-1,0]], img.shape)
    F = (np.conj(H)*G)/(np.abs(H)**2 + gamma*(np.abs(P)**2))
    return np.abs(np.fft.ifft2(F)).astype('uint8')

img = cv2.imread("image.jpg",0)
psf = np.ones((3,3))/9
out = clsf(img, psf, 0.01)
cv2.imwrite("clsf.jpg", out)
```

14) K-means Clustering

```
import cv2, numpy as np
img = cv2.imread("image.jpg")
data = img.reshape((-1,3)).astype(np.float32)
K = 3
_,labels,center = cv2.kmeans(
    data,K,None,
    (cv2.TERM_CRITERIA_EPS+cv2.TERM_CRITERIA_MAX_ITER,10,1.0),
    10,cv2.KMEANS_RANDOM_CENTERS)

center = np.uint8(center)
result = center[labels.flatten()].reshape(img.shape)
cv2.imwrite("cluster.jpg", result)
```