```
# code to make an image stylish by spreading horzontally mid of the image

import cv2
import math
import numpy as np
# capture frames from a camera
cap = cv2.VideoCapture(0)

# loop runs if capturing has been initialized.
while 1:

    # reads frames from a camera
    ret,img = cap.read()
    rows,cols,p = img.shape
    size = rows,cols,p
    y1 = np.zeros(size, dtype=np.uint8)     # y1 is defined for transferring pixel
from img
  # min. of the side length is specified to make square in image in later section
    m= int(cols/2)
    b = 0.008                      # b value is taken smaller b'coz exponent of it
may give large value
 # firstly image is divided in four eqaul parts
   #  to displace the pixel exponentially w.r.t y the function comes
    #                                         f:  x = a*(exp(b*y) - k)
 #  if middle of the image is (m,m) { for square }
#   for the image part located in side of (0, 0)
 #     y varies from 0 to m and for each y, x is varied from x1 to m to spread
uniformly into pixels from 0 to m,  x-wise
  #    exponentially x1 is varied from 0 to 5*m/6
   #     to find the value of a in f, putting x1 at (0,0) and (5m/6, m)
    #                               (x -> 0 to 5m/6)
    #                               (y -> 0 to m )
   #                         we get, 0 = a(exp(b*0)-k)  => k =1
  #                          and,    5m/6 = a(exp(b*m)-1)
 #                                => a = 5m/(6*(exp(b*m)-1))
#                            b has been taken by coder as 0.008 as per requirement
 #                           a and k has been calculated
  #          x1 will be calculated for each y after putting y in function f
#
# pixel is set in the position of spread in y1
    a = 5*m/(6*(math.exp(b*m) - 1))
    for y in range(0,m,1):
        x1 = a*(math.exp(b*y)-1) # x1 is calculated using function f
        x = x1
        while x<m:
                # pixels spreded from (m-x1) to m points
            k = (x-x1)*(m/(m-x1))   #k is the value of x-coordinate for spreaded
pixel considering uniform spreading
```

```
            y1.itemset((int(y),int(k),1),img[y,int(x),1])
            y1.itemset((int(y),int(k),2),img[y,int(x),2])
            y1.itemset((int(y),int(k),0),img[y,int(x),0])
            x = x+1
#respective coordinates for other quadrants are calculated using the mirror 
properties of first one
#and the same operation is done{ pixel transferred from img to y1 in spreaded 
form}"""
#


#    operation on the quadrant having coordinate (x=0,y=2m)

    for y in range(m,rows,1):
        x1 = a*(math.exp(b*(2*m-y))-1)  # (2*m-y) is the mirror image of y
        x = x1                          # x has the same range so no mirror image
        while x<m:
            k = (x-x1)*(m/(m-x1))
            y1.itemset((int(y),int(k),1),img[y,int(x),1])
            y1.itemset((int(y),int(k),2),img[y,int(x),2])
            y1.itemset((int(y),int(k),0),img[y,int(x),0])
            x = x+1
#    operation on the quadrant having coordinate (x=2m,y=0)

    for y in range(0,m,1):
        x1 = a*(math.exp(b*y)-1)
        x = 2*m-x1-1
        while x>m:
            k = 2*m -((2*m-x-x1)*(m/(m-x1)))
            y1.itemset((int(y),int(k),1),img[y,int(x),1])
            y1.itemset((int(y),int(k),2),img[y,int(x),2])
            y1.itemset((int(y),int(k),0),img[y,int(x),0])
            x = x-1
#    operation on the quadrant having coordinate (x=2m,y=2m)
#
    for y in range(m,rows,1):
        x1 = a*(math.exp(b*(2*m-y))-1)
        x = 2*m-x1-1
        while x>m:
            k = 2*m -((2*m-x-x1)*(m/(m-x1)))
            y1.itemset((int(y),int(k),1),img[y,int(x),1])
            y1.itemset((int(y),int(k),2),img[y,int(x),2])
            y1.itemset((int(y),int(k),0),img[y,int(x),0])
            x = x-1


    img = y1    # img equated to y1 and now is in spreaded form
   # Rest central black values{ [0,0,0] are eradicated by using simple interpolation
```

```
as done below
    for y in range(0,rows):
        x =0
        while x <cols:
            x1 = x
            while img[y,x].all() == 0 and x<= (cols-2) :
                img[y,x] = img[y,x-1]
                x = x+1
            if x> x1:
                x = x-1  # the extra increment in x inside while loop is removed
            x = x+1

    cv2.imshow('img',img)

    # Wait for Esc key to stop
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

# Close the window
cap.release()

# De-allocate any associated memory usage
cv2.destroyAllWindows()
```