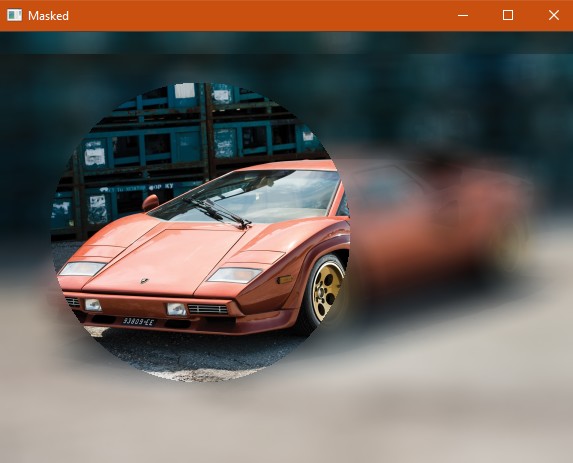
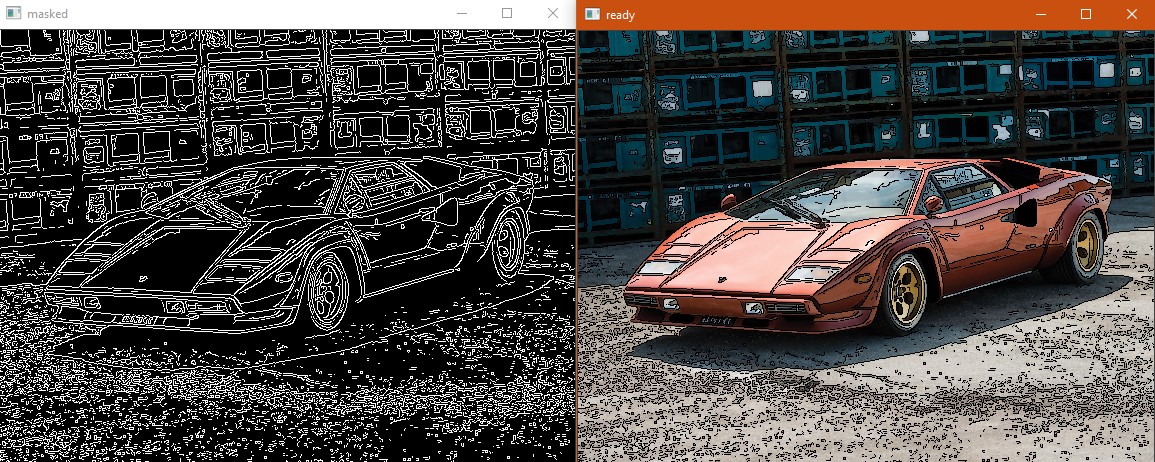
import os  
  
*# os.environ['KERAS\_BACKEND'] = "plaidml.keras.backend"  
# import keras  
  
  
  
# from tensorflow import keras*import pandas as pd  
import cv2 as cv  
import numpy as np  
import matplotlib.pyplot as plt  
path = **"1974-lamborghini-countach-classic-car-supercar-orange.jpg"**img = cv.imread(path)  
  
  
  
  
def resize (img, scale=0.3):  
 width = int(img.shape[1]\*scale)  
 height = int(img.shape[0]\*scale)  
 dimensions = (width, height)  
 return cv.resize(img, dimensions, interpolation=cv.INTER\_AREA)  
  
img = resize(img)  
*# cv.imshow(path,img)*

1. Сделать изображение размытым (или/и монохромным), кроме области внутри круга заданного радиуса, граница круга должна быть плавной.  
radius = 150  
  
bilateral= cv.bilateralFilter(img, 50 ,800,300)  
*# cv.imshow("bilateral",bilateral)*blank1 = np.zeros(img.shape[:2],dtype = **'uint8'**)  
  
mask = cv.circle(blank1, (200,200),radius,255,-1 )  
masked = cv.bitwise\_not(img,img,mask=mask)  
nottttt = cv.bitwise\_not(masked,bilateral,mask=mask)  
  
cv.imshow(**"Masked"**, nottttt)



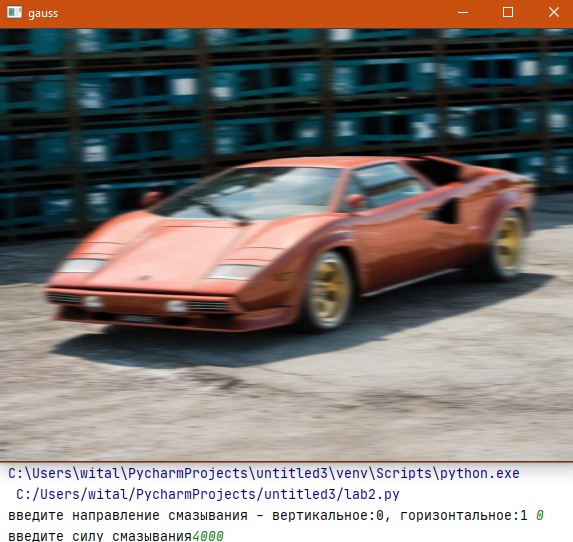
2. Выделить границы на изображении (canny) и применить размытие по гауссу ко всему изображению, кроме границ.  
gray = cv.cvtColor(img,cv.COLOR\_BGR2GRAY)  
  
canny = cv.Canny(gray, 125,100)  
  
gauss = cv.GaussianBlur(gray, (1111,1111), cv.BORDER\_DEFAULT)  
  
mask = canny  
masked = cv.bitwise\_not(gray,gray,mask=mask)  
  
nottttt = cv.bitwise\_not(masked,gauss,mask=mask)  
cv.imshow(**"final"**, nottttt)

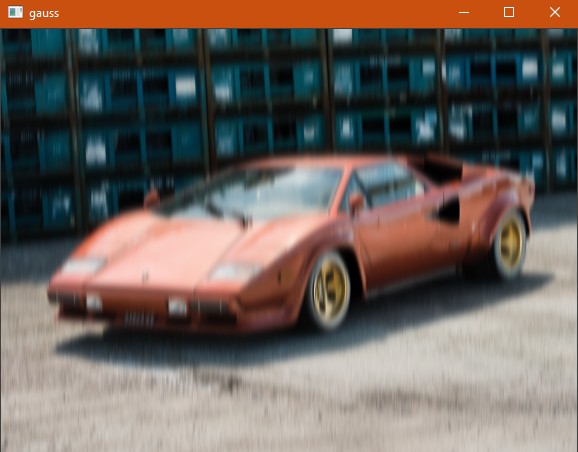


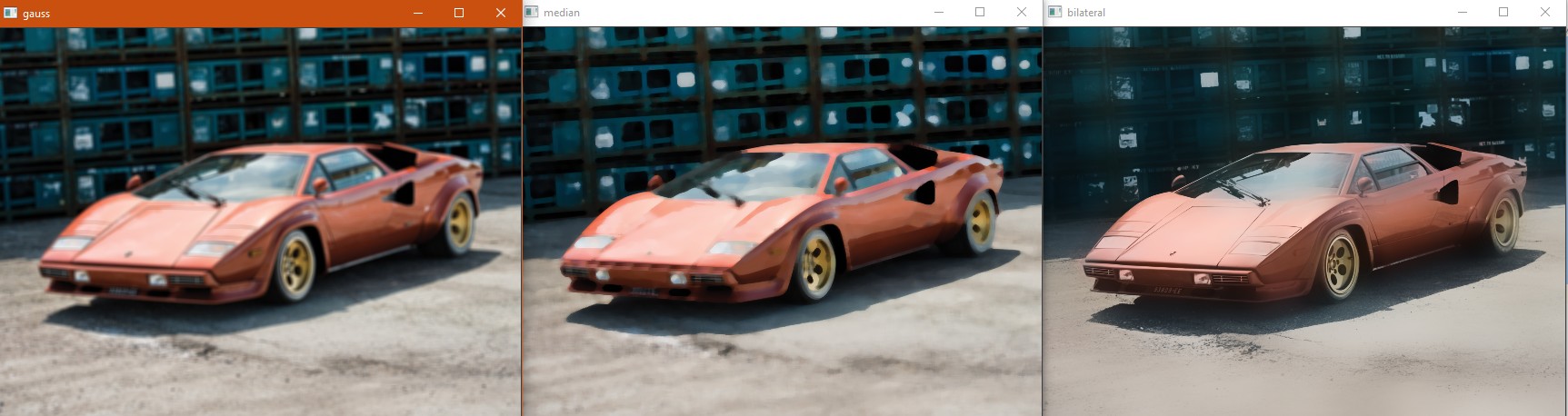
3. Разработать алгоритм, позволяющий выделять границы черным цветом, толщину линий и параметры их выделения (параметры для фильтра Кенни)canny = cv.Canny(img,125,100)  
new = img.copy()  
for y in range(new.shape[0]):  
 for x in range(new.shape[1]):  
 for c in range(new.shape[2]):  
 new[y,x,c] = np.clip(new[y,x,c], 0, 0)  
  
masked = cv.bitwise\_not(new,new,mask=canny)  
  
nottttt = cv.bitwise\_not(masked,img,mask=canny)  
  
cv.imshow(**"masked"**, masked)  
cv.imshow(**"ready"**, nottttt)  
  


4. Используя filter2D реализовать размытие с ядром в виде равномерно заполненного круга настраиваемого радиуса и кольца заданного радиуса и толщины, сравнить полученные изображения.radius = 70  
width = 20  
  
  
img\_src = img*#= cv.imread('sample.jpg')*kernel = np.array([[1,1,1,1,1,1,1,1,1],  
 [1,1,1,1,1,1,1,1,1],  
 [1,1,1,1,1,1,1,1,1],  
 [1,1,1,1,1,1,1,1,1],  
 [1,1,1,1,1,1,1,1,1],  
 [1,1,1,1,1,1,1,1,1],  
 [1,1,1,1,1,1,1,1,1],  
 [1,1,1,1,1,1,1,1,1],  
 [1,1,1,1,1,1,1,1,1]])  
  
kernel = kernel/(np.sum(kernel) if np.sum(kernel)!=0 else 1)  
  
img\_rst = cv.filter2D(img\_src,-1,kernel)  
  
blank1 = np.zeros(img.shape[:2],dtype = **'uint8'**)  
  
mask = cv.circle(blank1, (1100,750),radius,255,-1 )  
masked = cv.bitwise\_not(img,img,mask=mask)  
nottttt = cv.bitwise\_not(masked,img\_rst,mask=mask)  
cv.imshow(**'result.jpg'**,masked)  
  
mask1 = cv.circle(blank1.copy(), (650,900),radius,255,-1 )  
mask2 = cv.circle(blank1.copy(), (650,900),int(radius-width),255,-1 )  
masked2 = cv.bitwise\_xor(mask1,mask2)  
  
masked3 = cv.bitwise\_not(img,img,mask=masked2)  
nottttt3 = cv.bitwise\_not(masked3,img\_rst,mask=masked2)  
  
  
cv.imshow(**'round.jpg'**,nottttt3)  
  

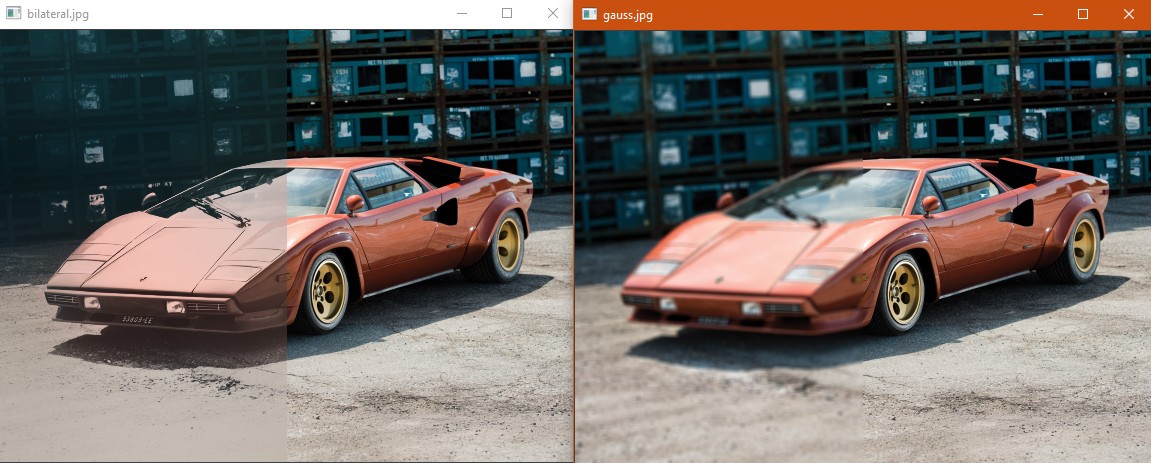

5. Реализовать эффект смазывания от движения (motion blur) с настраиваемым направлением движения и степенью смазыванияtry:  
 vector = int(input(**"введите направление смазывания - вертикальное:0, горизонтальное:1 "**))  
 if vector!= 1 or vector!=0:  
 vector = 1  
 force = int(input(**"введите силу смазывания"**))  
 if force%2==0:  
 force=force+3  
except:  
 vector = 0  
 force = 4003  
  
  
if vector ==0:  
 kortej = (1,force)  
else:  
 kortej = (force, 1)  
  
gauss = cv.GaussianBlur(img, kortej, cv.BORDER\_DEFAULT)  
cv.imshow(**"gauss"**,gauss)



**

6. Сравнить фильтрацию по Гауссу, билатеральную и медианную фильтрацию (реализовать программу, выводящую результат применения разных фильтров одновременно, сравнить их для разных значений параметров)gauss = cv.GaussianBlur(img, (5,5), cv.BORDER\_DEFAULT)  
cv.imshow(**"gauss"**,gauss)  
  
median = cv.medianBlur(img,7)  
cv.imshow(**"median"**,median)  
  
bilateral= cv.bilateralFilter(img, 50 ,140,150)  
*# второй - диаметор . 3 - цвет сигмы- больше значение - больше разных цветов будет влиять на пиксель. сигма спейс- расстояние влияния*cv.imshow(**"bilateral"**,bilateral)  
  
  


7. Сравнить эффект от применения фильтрации по Гауссу и билатеральной фильтрации (продемонстрировать поведение фильтров на границах для различных значений параметров фильтра)blank1 = np.zeros(img.shape[:2],dtype = **'uint8'**)  
gauss = cv.GaussianBlur(img.copy(), (5,5), cv.BORDER\_DEFAULT)  
  
mask = cv.rectangle(blank1.copy(),(img.shape[1]//2,0),(img.shape[1] ,img.shape[0]),254,-1)  
masked = cv.bitwise\_not(img.copy(),img.copy(),mask=mask)  
result = cv.bitwise\_not(masked,gauss,mask=mask)  
cv.imshow(**'gauss.jpg'**,result)  
  
bilateral= cv.bilateralFilter(img.copy(), 150 ,150,150)  
mask2 = cv.rectangle(blank1.copy(),(img.shape[1]//2,0),(img.shape[1] ,img.shape[0]),254,-1)  
masked2 = cv.bitwise\_not(img.copy(),img.copy(),mask=mask2)  
result2 = cv.bitwise\_not(masked2,bilateral,mask=mask2)  
cv.imshow(**'bilateral.jpg'**,result2)



8. Используя фильтрацию по гауссу (либо билатеральный фильтр) выполнить многоуровневую фильтрацию (усилить детали изображения с размерами порядка 5-10 пикселей, оставив неизменными более мелкие детали)blank1 = np.zeros(img.shape[:2],dtype = **'uint8'**)  
  
bilateral= cv.bilateralFilter(img, 200 ,150,100)  
bilateral2= cv.bilateralFilter(bilateral, 200 ,150,100)  
bilateral3= cv.bilateralFilter(bilateral2, 200 ,150,100)  
  
mask = cv.rectangle(blank1.copy(),(img.shape[1]//2,0),(img.shape[1] ,img.shape[0]),254,-1)  
masked = cv.bitwise\_not(img.copy(),img.copy(),mask=mask)  
result = cv.bitwise\_not(masked,bilateral3,mask=mask)  
cv.imshow(**'gauss.jpg'**,result)  
  
  
