Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

     Факультет информационных технологий и управления

     Кафедра интеллектуальных информационных технологий

ОТЧЁТ

по лабораторной работе №4

по дисциплине

ОБРАБОТКА ИЗОБРАЖЕНИЙ В ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ

Выполнил: Рублевская Е. А.

Гр. 321702

Проверил: Сальников Д. А.

Минск 2025

**Цель:**

Реализовать программно сегментацию посредством выделения границ областей или посредством разметки точек области.

**Ход работы:**

1. Загружаем исходное изображение для дальнейшей работы.
2. Применяем функцию для Гауссового размытия, чтобы уменьшить шум изображения и сделать границы изображения более четкими.
3. С помощью функции оператора Канни вычисляем градиенты, нормализуем его для того, чтобы интерпретировать значения как интенсивность границ.
4. Выводим входное и обработанное изображения.

**Средства разработки:**

* NumPy – библиотека, которая используется для создания и работы с фильтрами Собеля и Гауссовым размытием, выполнения математических операций.
* OpenCV – библиотека для компьютерного зрения, обработки изображения и видео. Используется для загрузки изображения и применения фильтров.
* Matplotlib – библиотека, которая используется для отображения готовых результатов.

**Основные части кода:**

* Загрузка исходного изображения:

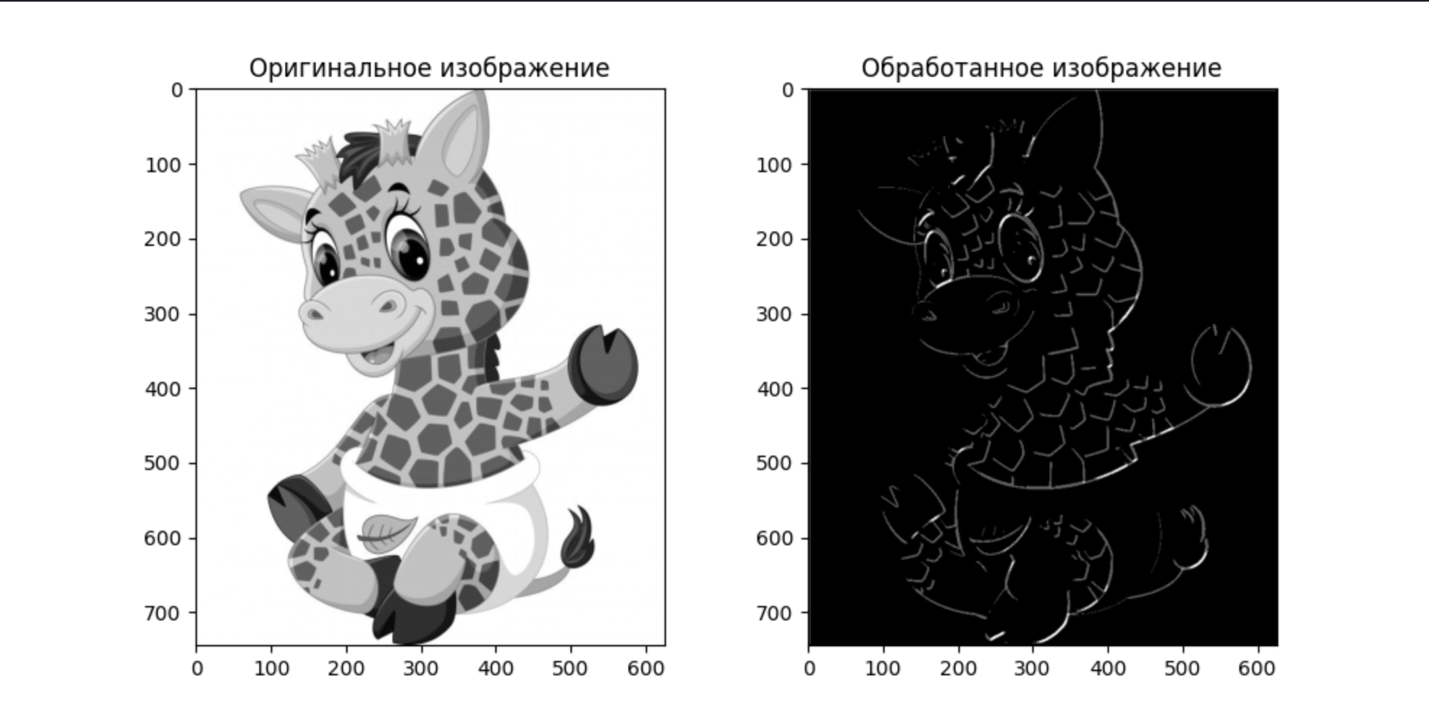
image = cv2.imread('giraffe.jpg'**,** cv2.IMREAD\_GRAYSCALE)

* Применение Гауссового размытия. Данная функция создает матрицу, которая определяет, как каждый пиксель будет размываться в зависимости от его соседей. Чем ближе пиксель к центру, тем больше его вес. Функция проходит по каждому пикселю изображения и заменяет его значением, вычисленным с использованием Гауссового фильтра, чтобы уменьшить шум и сделать границы более четкими.

def custom\_gaussian\_blur(image**,** kernel\_size**,** sigma):  
 kernel = np.zeros((kernel\_size**,** kernel\_size)**,** np.float32)  
 half\_size = kernel\_size // 2  
 for x in range(-half\_size**,** half\_size + 1):  
 for y in range(-half\_size**,** half\_size + 1):  
 kernel[x + half\_size**,** y + half\_size] = (1 / (2 \* np.pi \* sigma \*\* 2)) \* np.exp(  
 -(x \*\* 2 + y \*\* 2) / (2 \* sigma \*\* 2))  
 kernel /= kernel.sum()  
  
 # Применение фильтра  
 height**,** width = image.shape  
 blurred = np.zeros\_like(image**,** dtype=np.float32)  
 for i in range(half\_size**,** height - half\_size):  
 for j in range(half\_size**,** width - half\_size):  
 region = image[i - half\_size:i + half\_size + 1**,** j - half\_size:j + half\_size + 1]  
 blurred[i**,** j] = np.sum(region \* kernel)  
  
 return blurred.astype(np.uint8)

* Функция оператора Канни. Мы используем cv2.filter2D(), чтобы применить эти фильтры для изображения и получить градиенты gradient\_x и gradient\_y. Затем мы вычисляем величину и направление градиента по формуле гипотенузы. После этого величины градиентов нормализуются от 0 до 255, чтобы их можно было интерпретировать как интенсивность границ. Если величина градиента выше верхнего порога, такие пиксели считаются сильными границами и устанавливаются как 255 (белые). Если величина градиента находится между нижним и верхним порогом, пиксели помечаются как слабые границы и устанавливаются как 100. Все остальные пиксели остаются черными, т. е. 0.
* def custom\_canny(image**,** low\_threshold**,** high\_threshold):  
   # Градиенты с помощью Собеля  
   sobel\_x = np.array([[-1**,** 0**,** 1]**,** [-2**,** 0**,** 2]**,** [-1**,** 0**,** 1]])  
   sobel\_y = np.array([[-1**,** -2**,** -1]**,** [0**,** 0**,** 0]**,** [1**,** 2**,** 1]])  
    
   gradient\_x = cv2.filter2D(image**,** -1**,** sobel\_x)  
   gradient\_y = cv2.filter2D(image**,** -1**,** sobel\_y)  
    
   magnitude = np.hypot(gradient\_x**,** gradient\_y)  
   direction = np.arctan2(gradient\_y**,** gradient\_x)  
    
   # Нормализация градиентов  
   magnitude = magnitude / magnitude.max() \* 255  
   magnitude = magnitude.astype(np.uint8)  
    
   # Определение порогов  
   edges = np.zeros\_like(magnitude**,** dtype=np.uint8)  
   strong = magnitude > high\_threshold  
   weak = (magnitude >= low\_threshold) & (magnitude <= high\_threshold)  
    
   edges[strong] = 255  
   edges[weak] = 100 # Выделение слабых границ для дальнейшей обработки  
    
   return edges
* Отображение результатов:
* plt.figure(figsize=(10**,** 5))  
  plt.subplot(1**,** 2**,** 1)  
  plt.title('Оригинальное изображение')  
  plt.imshow(image**,** cmap='gray')  
    
  plt.subplot(1**,** 2**,** 2)  
  plt.title('Обработанное изображение')  
  plt.imshow(edges**,** cmap='gray')  
  plt.show()

**Результат работы программы:**

****

**Вывод:**

В ходе данной лабораторной работы была разработана программа, которая выполняет сегментацию изображения через выделение границ с использованием оператора Канни. Она включает в себя этапы размытия, вычисления градиентов и пороговой обработки для выделения границ.

В результате была разобрана сегментация изображений – процесс разделения изображений на несколько значимых частей или областей с целью упрощения его анализа и интерпретации.

Основными целями сегментации являются:

* *Выделение объектов.* Сегментация помогает отделить объекты на изображении от фона, что упрощает их анализ и обработку.
* *Анализ форм и размеров.* Разделение изображения на отдельные объекты позволяет изучать их форму, размер, ориентацию и другие характеристики.
* *Извлечение признаков.* Сегментированные области могут использоваться для дальнейшего извлечения признаков, например, цвет, текстура, контуры, которые важны для распознавания объектов.
* *Сокращение объема данных.* Сегментация помогает свести изображение к более простому набору данных, что облегчает его обработку и снижает вычислительные затраты.