**Поиск цветных объектов на видео с помощью OpenCV-Python.**

Нейросеточка: Привет! Ваня, помнишь, мы с тобой искали цветные объекты на фотографиях?

Иван: Да, помню. Было интересно!

Нейросеточка: А ты знаешь, что поиск цветных объектов — это базовая задача в системах с компьютерным зрением? Такие задачи решают при разработке автономных систем управления беспилотных автомобилей. Например, чтобы автомобиль мог распознавать дорожные знаки или сигналы светофора.

Иван: Ого, круто! А почему же тогда у нас получалось находить не только нужный объект, но и много других мелких крапинок?

Нейросеточка: Потому что мы искали по всему кадру. Но ведь искомый объект может быть только в определённой его части кадра. Поэтому нужно искать внутри границ этой области. Тогда будет меньше лишних крапинок.

Иван: Это как будто ищешь конфету в коробке, а не просто смотришь на всё подряд.

Нейросеточка: Точно! И тогда вероятность найти нужный объект гораздо выше. Но сначала нужно определить область поиска. Мы не можем сразу указать нужную область поиска, нам придется действовать опытным путем.

Иван: А что это значит?

Нейросеточка: Это значит, что мы будем пробовать разные размеры и смотреть, какой результат получается. И выберем тот, который лучше всего подходит для нашей задачи.

Иван: Звучит сложновато. Но я готов попробовать!

Нейросеточка: Отлично! Тогда начнём с программы, которую мы уже использовали для поиска объекта в цветовом HSV-пространстве.

Рассказчик: Порядок действий для поиска цветного объекта в видеопотоке будет состоять из нескольких этапов::

1. Открытие видеофайла
2. Преобразование изображения каждого кадра в цветовое пространство HSV.
3. Обрезка кадра по размерам видимой области с искомым объектом.
4. Опредление границ цветового диапазона для создания цветовой маски.
5. Применение HSV-маски к каждому кадру видеопотока после его преобразования в HSV-пространство.
6. Деомнстрация результатов наложения цветовой HSV-маски в отдельном окне.

Нейросеточка: давай напишем небольшую программу, которая реализует этот алгоритм. Мы будем находить объекты желтого цвета в видеопотоке из файла, полученного с автомобильного видеорегистратора. Для программы нам пригодится библиотека NumPy, которая используется для оптимизации сложных вычислений. Она предоставляет множество функций для работы с данными, таких как сложение, умножение и многое другое.

Иван: А какое отношение NumPy имеет к openCV ?

Нейросеточка: OpenCV использует NumPy для работы с массивами данных, например, пиксельными изображениями

Рассказчик: Инструкция по установке NumPy находится в материалах к занятию.

Иван: Давай уже будем писать программу.

Нейросеточка:

Сначала выбери видео, которое нужно обработать.

Иван: Ой, а у меня нет видео.

Нейросеточка: Ничего страшного, ты можешь попросить нейросеть сгенерировать небольшое видео.

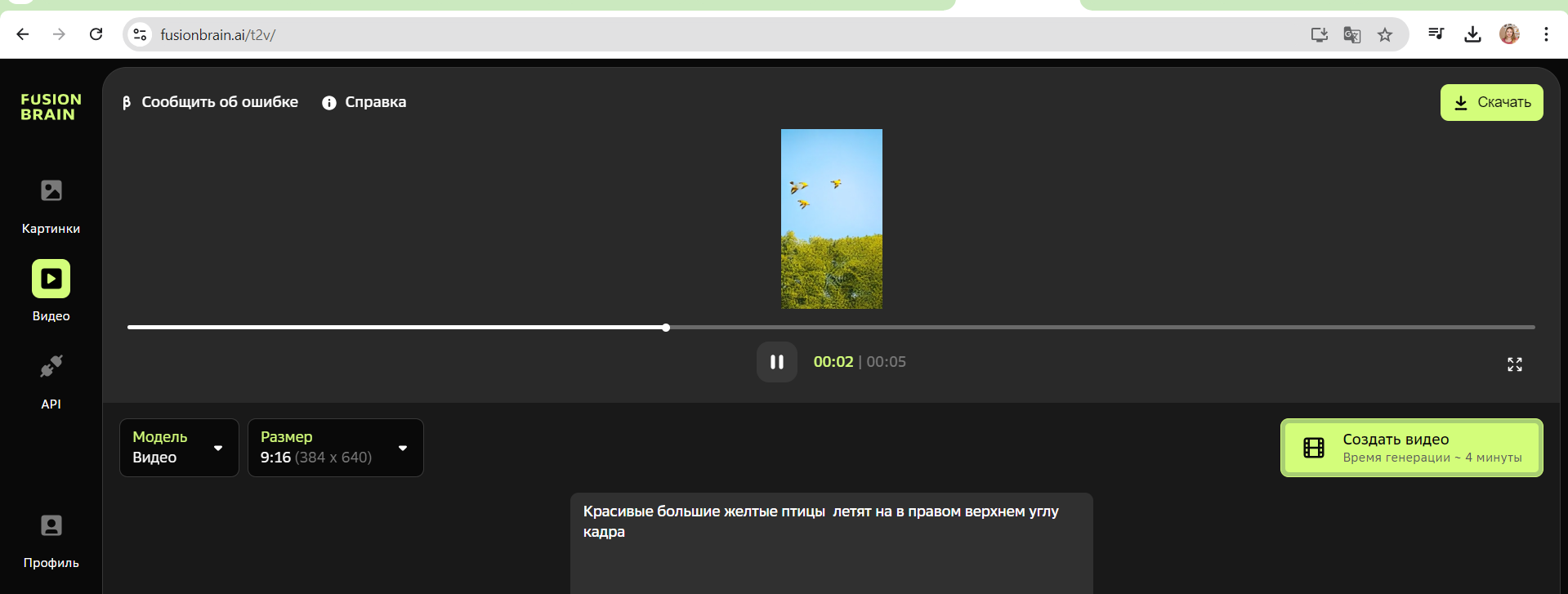
Рассказчик: Нейросети уже умеют создавать реалистичные видео-анимации, которые используются игровой индустрии или кино. Так, например, российская нейронная сеть Kandinsky Video умеет создавать небольшие видео. Воспользоваться ей можно двумя способами: на сайте нейросети Fusion Brain или через телеграм-бота. Подробная инструкция находится в материалах к занятию.

Иван: Я хочу видео с птицами на фоне неба.

Нейросеточка: Переходи на сайт <https://fusionbrain.ai/t2v/>. Выбери вкладку Видео на панели слева. В поле с запросом введи свой текст, установи необходимый размер и выбери модель. Это может быть анимация или видео.

Иван: Хорошо. Ввожу текст: Красивые большие желтые птицы летят над деревьями на фоне голубого неба. Нажимаю Создать видео. Время генерации 4 минуты. Подождем.

Рассказчик: После генерации видео нужно нажать на кнопку Скачать, расположенную в правом верхнем углу.

для дизайна, нужно будет отрисовать вот так: 

Нейросеточка: Иван, давай пока начнем писать Иван: Думаю, что сначала надо подключить библиотеку CV и открыть видеофайл?

Нейросеточка: Еще не забудь подключить библиотеку numpy.

Иван: Точно!

*На экране окно VSC. Иван вводит фрагмент кода.*

*import numpy as np*

*import cv2*

*capImg = cv2.VideoCapture('video1.avi')*



Нейросеточка: Далее нужно воспользоваться циклом while для непрерывного считывания кадров из видеопотока до тех пор, пока не будет достигнут конец видео. Внутри цикла выполняются следующие шаги:

1. Считывание кадра из видео потока с помощью метода capImg.read()
2. Проверка не достигнут ли конец видео.
3. Если кадр успешно прочитан то выполняется обработка кадра.
4. Если кадр не прочитан, то цикл завершается.

Цикл завершится когда будет достигнут конец видеопотока..

Иван: И как написать программу для этого?

Нейросеточка: Иван, давай я тебе покажу этот фрагмент кода. А ты потом попросишь у Гигачата объяснить его построчно.

Иван: Хорошо

*Нейросеточка на экран выводит фрагмент:*

while capImg.isOpened():

ret, frame = capImg.read()

if not ret or frame is None:

break

frame\_hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

height, width = frame.shape[:2]

crop\_frame = frame[0:height, 0:width]

crop\_frame\_hsv = frame\_hsv[0:height, 0:width]

low\_Yellow = np.array([20, 100, 100], dtype = "uint8")

high\_Yellow = np.array([30, 255, 255], dtype = "uint8")

yellow\_mask = cv2.inRange(crop\_frame\_hsv, low\_Yellow, high\_Yellow)

cv2.imshow("video\_mask", yellow\_mask)

cv2.imshow("video\_frame", crop\_frame)

key\_press = cv2.waitKey(30)

if key\_press == ord('q'):

break



Нейросеточка: после цикла тебе нужно будет прописать две строки: Одна - освобождает ресурсы, занятые видеопотоком, что позволяет программе корректно закрыть видеофайл и освободить память. И вторая закрывает все окна созданные OpenCV.

*На экране в это время дописываются еще 2 строки внизу кода:*

capImg.release()

cv2.destroyAllWindows()



*Для дизайнеров: Полный код программы будет выглядеть так:*

import numpy as np

import cv2

# загрузка видео файла

capImg = cv2.VideoCapture('video1.avi')

while capImg.isOpened():

ret, frame = capImg.read()

if not ret or frame is None:

break

# конвертация кадра в HSV

frame\_hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

# опредеения размера рамки

height, width = frame.shape[:2]

# обрезка рамки до исходных размеров

crop\_frame = frame[0:height, 0:width]

crop\_frame\_hsv = frame\_hsv[0:height, 0:width]

# определение желтой цветовой гаммы

low\_Yellow = np.array([20, 100, 100], dtype = "uint8")

high\_Yellow = np.array([30, 255, 255], dtype = "uint8")

# создание маски желтого цвета

yellow\_mask = cv2.inRange(crop\_frame\_hsv, low\_Yellow, high\_Yellow)

# отображение кадров

cv2.imshow("video\_mask", yellow\_mask)

cv2.imshow("video\_frame", crop\_frame)

# выход из программы при нажатии клавиши 'q'

key\_press = cv2.waitKey(30)

if key\_press == ord('q'):

break

capImg.release()

cv2.destroyAllWindows()



Иван: Сложно. Надо попросить гигачат рассказать, что означает каждая строка в цикле.

Рассказчик:

Промпт: Объясни построчно назначение каждой строки фрагмента кода:

while capImg.isOpened():

ret, frame = capImg.read()

if not ret or frame is None:

break

frame\_hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

height, width = frame.shape[:2]

crop\_frame = frame[0:height, 0:width]

crop\_frame\_hsv = frame\_hsv[0:height, 0:width]

low\_Yellow = np.array([20, 100, 100], dtype = "uint8")

high\_Yellow = np.array([30, 255, 255], dtype = "uint8")

yellow\_mask = cv2.inRange(crop\_frame\_hsv, low\_Yellow, high\_Yellow)

cv2.imshow("video\_mask", yellow\_mask)

cv2.imshow("video\_frame", crop\_frame)

key\_press = cv2.waitKey(30)

if key\_press == ord('q'):

break

Результат:   
1. Открывается видеопоток capImg.

2. Проверяется, открыт ли видеопоток. Если да, продолжаем выполнение кода, если нет - выходим из цикла.

3. Считывается кадр из видеопотока capImg.

4. Проверяется успешность считывания кадра. Если неудачно или кадр пустой, цикл прерывается.

5. Преобразуется цветовая схема кадра в HSV (от BGR).

6. Получается ширина и высота кадра.

7. Создается отдельная область crop\_frame внутри фрейма с аналогичными размерами.

8. Создается отдельная область crop\_frame\_hsv внутри фрейма после преобразования цветовой схемы.

9. Создаются нижняя и верхняя границы цветового фильтра для желтого цвета.

10. Применяется фильтр по цвету кадра в цветовом пространстве HSV и сохраняется в переменную yellow\_mask.

11. Отображается изображение маски желтого цвета.

12. Отображается обрезанный кадр crop\_frame.

13. Ожидается нажатие клавиши с клавиатуры.

14. Если нажата клавиша 'q', цикл прерывается.

~~Иван: Интересно, что такое фрейм.~~

~~Промпт: Объясни, что такое фрейм в этом коде.~~

*~~Для дизайнеров: Этот запрос должен быть следом на предидущим на экране, как продолжение диалога с нейросетью, а не новый.~~*

~~Рассказчик: Результат:~~

~~В данном коде фрейм — это отдельный кадр из видеопотока, используемый для обработки и анализа изображения, включая применение цветовых фильтров и выделение объектов.~~

Иван: Теперь мне стало понятно! Запускаю программу. На экране появляется картинка.

Нейросеточка: Теперь ты умеешь искать цветные объекты. Подбери границы цветового диапазона для зеленых объектов на видео. И напиши программу, которая будет определять объекты зеленого цвета в видеопотоке из файла, с которым мы только что работали.

Сегодня на занятии вы:

Узнали, что для чего нужна бибилитека numpy

Составили программу, которая позволяет искать цветные объекты на видео.