

# **1 Введение:**

## **Аннотация:**

В этой работе будет продемонстрирован морфологический анализ изображений, а именно поиск отличий между изображениями состоящими из областей постоянного цвета. Изображения будут сегментированы различными методами и среди них, при помощи сравнения с сегментацией выполненной человеком, будет выбран наилучший способ.

## **Актуальность:**

Методы морфологического анализа изображений представляют собой раздел компьютерного зрения, а именно математическую интерпретацию изображений. Особый интерес вызывает подход Пытьева к сравнению изображений, который демонстрирует устойчивость к некачественной сегментации, а результатом является визуально понятное человеку отличие между изображениями. Также актуальным является вопрос сегментации изображений. В настоящее время не существует модели которая могла бы выдавать универсальную и подходящую для всех сфер сегментацию. В этой работе будут приведены 4 метода сегментации 2 обычных и 2 нейросетевых.

## **Цель работы**

Целью работы является демонстрация различных методов сегментации и анализ этих методов при помощи морфологического анализа изображений.

## **Решаемые задачи:**

1. Реализация различных методов по сегментации цветных изображений на области постоянного цвета.
2. Анализ и выделение лучшего метода сегментации при поиске отличий между изображениями

## **2 Теоретическая часть работы**

Основные задачи и математическая модель которую мы будем использовать для изображений

### **2.1 Математическая Модель цветного изображения:**

Основные представления о модели, и о том, что является изображением с точки зрения математики и как мы будем искать отличия между ними по форме.

### 2.1.1 Цветное изображение в математике

Цветное изображение представимо в виде математического объекта

$$f(x) = \sum_{j=1}^N a_j(x), \bar{\phi}_j \quad (1)$$

где  $x$ - координата геометрического пространства изображения( $x, y, 2d$ ),  $N$ -количество цветовых векторов,  $a_j$ -яркость вектора,  $\bar{\phi}$ - базисный цветовой вектор.

В вопросах компьютерного зрения превалирует система rgb для хранения цветов изображений, но в данной работе будет использована hsv система для хранения векторов цвета.

Тогда наше изображение можно записать как :

$$f(x) = V(x)\bar{\phi}_v + H(x)\bar{\phi}_H + S(x)\bar{\phi}_S \quad (2)$$

Также HSV цветное пространство удобно интерпретировать в цилиндрической системе координат к примеру:

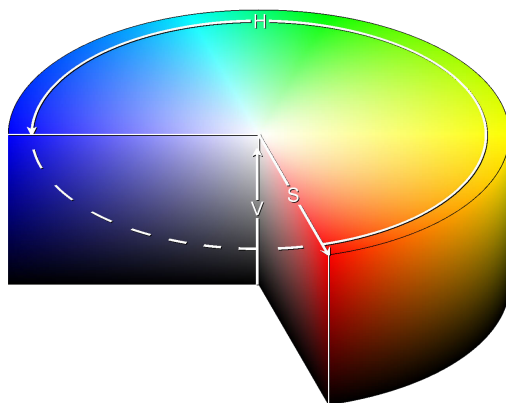


Рис.1 HSV цветное пространство визуальная интерпретация

Сегмент изображения это  $A$  набор координат  $x$  изображения выбранных по какому либо правилу и сегмент изображения задается как

$$S(f, A) = f(x) * \chi(x, A) \quad (3)$$

где  $\chi$  это принадлежность к сегменту.

$$\chi(x, A) := \begin{cases} 1, & x \in A_i, \\ 0, & x \notin A_i. \end{cases}$$

### 2.1.2 Поиск отличий по форме

Введем такое значение как пересечение двух сегментов

$$I(A_i, B_j) = \chi(x, A_i) * \chi(x, B_j) \quad (4)$$

где  $A_i, B_j$ -сегменты изображений  $a, b$

Тогда отличия между изображениями по форме задается как:

$$\Delta(x)_{ij} = \sum_{i=0}^N \sum_{j=0}^M \left[ I(A_i, B_j) * (f_A(x) - f_B(x)) + \right. \\ \left. + \chi(x, A_i) * (f_A(x) - f_B(x)) * \sum I(A_i, B_j) / \sum \chi(A_i) \right] \quad (5)$$

где  $\Delta(x)_{ij}$ -отличие между двумя сегментами.

Новое изображение отличия всех сегментов изображения записывается как:

$$f'(x) = \sum_{i=0}^N \sum_{j=0}^M \Delta(x)_{ij} \quad (6)$$

такой подход позволяет наглядно показать отличие между формами изображений.

### 2.1.3 Ортогональная проекция

Преобразование изображений позволяющее получить минимальное расстояние между ними по яркости и контрастности.

для этого посчитаем среднее значение по параметрам s,v в hsv пространстве и сделаем проекцию по яркости и контрасту изображения 1 на изображение 2 ;

$$H'_{f1}(x) = H_{f1}(x) * \frac{\sum_{i=0}^N H_{f2}(x_i) * M}{\sum_{j=0}^M H_{f1}(x_j) * N} \quad (7)$$

аналогично с V.

## 2.2 Методы выделение областей постоянного цвета

В данной работе использованы следующие методы :

1. Каждый пиксель изображения является областью постоянного цвета.
2. Сегментация изображения на области методом суперпикселей.
3. Сегментации на основе человеческого зрения для валидации.
4. Свёрточная нейронная сеть обученная на поиск границ областей постоянного цвета.
5. Свёрточная нейронная сеть обученная для поиска областей постоянного цвета близких к цветовым векторам из заранее заданных.

### 2.2.1 Методы без нейронных сетей

#### 2.2.1.1 Пиксели

Каждый пиксель изображения является сегментом, так что результатом поиска отличий будет просто вычитание одного изображения из другого.

#### 2.2.1.2 Суперпиксели

**Задача** : реализовать алгоритм который получает на вход изображение, а возвращает набор сегментов изображения.

**Суперпиксели** — это компактные группы пикселей, сохраняющие границы объектов на изображении. Они упрощают обработку, сокращая вычислительные затраты, и используются в задачах. В данной работе используется метод кластеризации изображения суперпикселями SLIC.

### SLIC (Simple Linear Iterative Clustering)

Наиболее популярный метод, основанный на кластеризации **k-means** в 5D-пространстве (цвет LAB + координаты XY).

Алгоритм состоит из следующих шагов:

1 Инициализация центроидов:

$$C_k = [l_k, a_k, b_k, x_k, y_k] \text{ с шагом } S = \sqrt{\frac{N}{K}}.$$

Для каждого пикселя вычисляется расстояние

$$D = \sqrt{d_{lab}^2 + \left(\frac{m}{S}\right)^2 d_{xy}^2} \quad (8)$$

$d_{lab}$ - цветовое расстояние

$d_{xy}$ - геометрическое расстояние

$m$ - параметр компактности

$N$  - число пикселей,

$K$  - число суперпикселей.

Новый центроид считается таким образом:

$$C_k^{\text{new}} = \frac{1}{|S_k|} \sum_{i \in S_k} \begin{bmatrix} l_i \\ a_i \\ b_i \\ x_i \\ y_i \end{bmatrix}. \quad (9)$$

Критерий остановки алгоритма:

$$\|C_k^{\text{new}} - C_k^{\text{old}}\| < \epsilon. \quad (10)$$

### 2.2.1.3 Человеческое зрение

Выделение областей постоянного цвета по субъективным критериями основными из которых является семантика + цвет объектов на изображении.

### 2.2.1.4 Метод К-средних

Наиболее популярный метод кластеризации для данных. Заранее задается количество кластеров  $K$  и выбираем такие их значения чтобы минимизировать функционал:

$$V = \sum_{i=1}^k \sum_{x \in A} (x - \mu_i)^2$$

### 2.2.2 Нейронные сети

В этой работе для обучении обеих моделей будем использовать архитектуру свёрточной нейронной сети UNET.

В оригинальной, авторской статье 2015 года UNET модель применялась для сегментации медицинских изображений, но модель также нашла применение и для сегментации других изображений. Несмотря на давность релиза модели она все еще показывает хорошие результаты и на основе ее архитектуры создаются новые улучшенные версии.

В нашем случае мы возьмем ее классическую архитектуру из первой статьи и обучим модели.

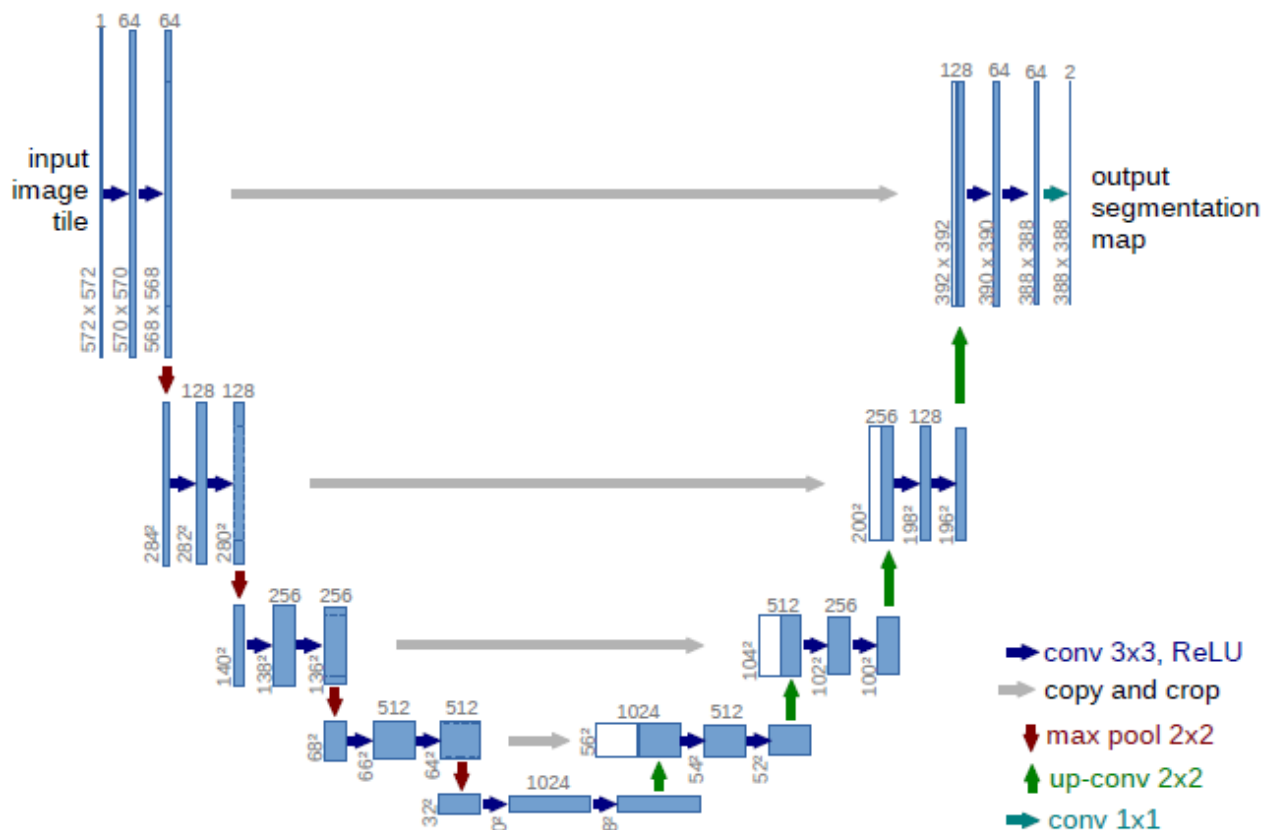


Рис. 2 Архитектура модели Unet

### 2.2.2.1 Модель обученная на поиск границ

**Задача :** Получить модель, получающую на входе изображение и возвращающую бинарную маску с размеченными границами сегментов постоянного цвета.

**Основная идея:**

Получив на выходе маску с границами инвертировать её, и получить области примерно постоянного цвета. После чего выделить поиском отдельные сегменты и уже оперировать с ними, а также из-за довольно таки большой толщины границ их тоже выделить как сегменты, но выбирать в сегмент не все что находится в границе а только в пределах некоторого заранее заданного расстояния.

### Обучающий датасет:

Для обучения модели сгенерируем синтетический датасет состоящий из примитивов различного цвета, поверх которых добавлен шум по гауссу в качестве входных данных, а таргетами будет бинарная маска где выделена граница примитивов. Так же при обучении будут использоваться линейные преобразования как: поворот, смещение, и сжатие, и расширения, изображения для лучшего обучения.

В качестве функции потерь используем Dixeloss

$$DiceLoss(y, p) = 1 - \frac{2 * y * p + \epsilon}{y + p + \epsilon}$$

$\epsilon$ - малая поправка в работе=0.001



Рис.3 Пример изображения из обучающего датасета

#### 2.2.2.2 Модель обученная на поиск областей постоянного цвета

**Задача :** Получить модель , получающую на вход изображение и возвращающую маску с классами к которым принадлежат пиксели и чтобы это были области постоянного цвета.

##### Основная идея

Выделить конечное количество векторов и сегментировать изображение на области постоянного цвета где области будут иметь цвет одного из списка цветов. В этой работе я выделил 32 конечных вектора (27 цветных и 5 оттенков серого). По причине того что конечных цветовых векторов не так много, на выходе мы получим не слившиеся в один, отдельные области примерно постоянного цвета, при этом будет не так много отдельных областей на предметах которые можно было бы назвать областью одного постоянного цвета.



Рис.4 Заранее выделенные 32 цвета

### Обучающий датасет:

Для обучения этой модели сгенерируем синтетический датасет, состоящий из примитивов(многоугольники, эллипсы , звезды), поверх которых добавим шум по гауссу и градиент по SV(Яркость и контраст) координатам вектора цвета. Таргетом будет маска где на месте примитива будут находиться метки класса ближайшего к исходному вектору цвета.

Функция потерь dice loss для нескольких классов

$$DiceLoss(y, p) = 1 - 2 * \frac{\sum_l \sum_i (y_i^l * p_i^l) + \epsilon}{\sum_l \sum_i (y_i^l + p_i^l) + \epsilon}$$

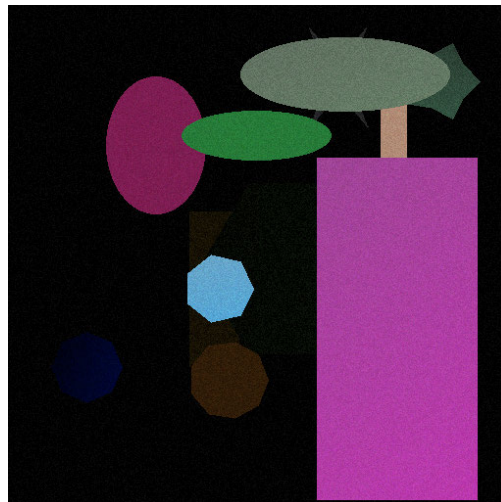


Рис. 5 Пример изображения из обучающего датасета

## 3. Реализация методов

Был использован язык программирования python, а так же ряд библиотек (numpy, albumentations, torch, cv2, PIL). Так же для обучения модели была использована видеокарта с поддержкой cuda. Структура модели unet взята из оригинальной статьи.

### 3.1 Результат сегментации

#### 3.1.1 Модель обученная на поиск границ

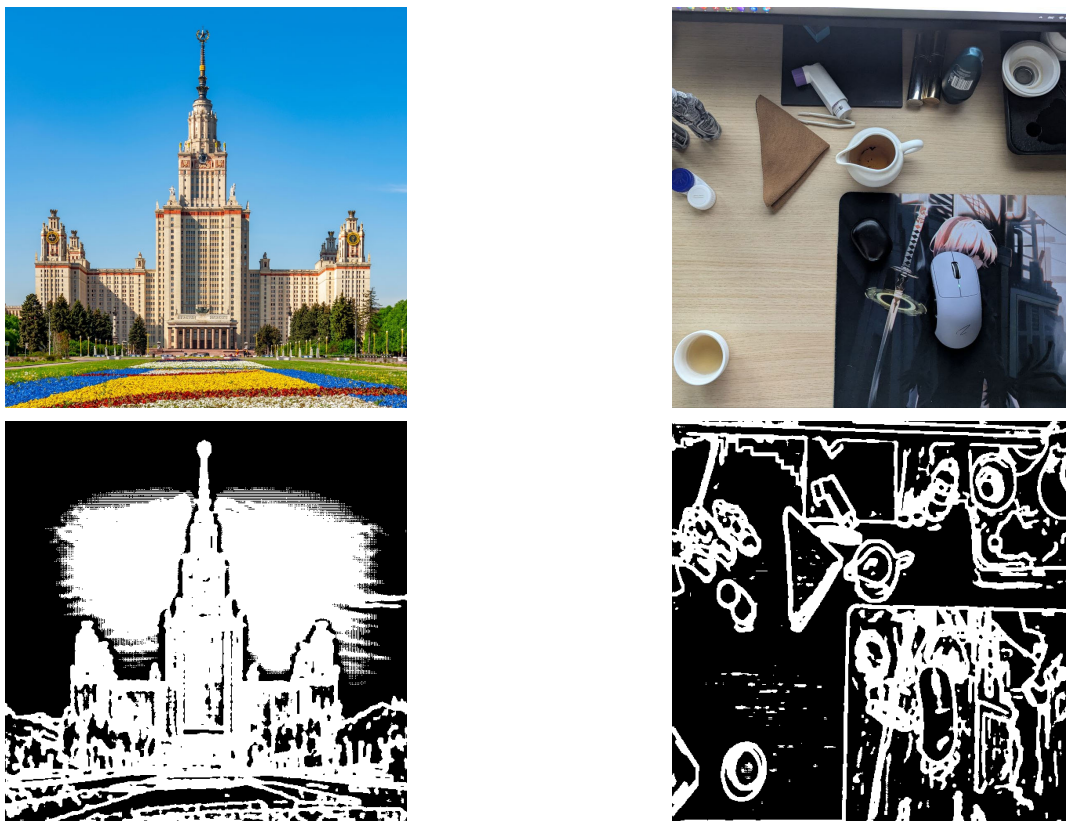


Рис. 6 Пример работы модели

Как видно из результатов сложные изображения оно плохо сегментирует, с более простыми сценами модель справляется лучше, но для нашей цели выделяет немного лишние границы.

### 3.1.2 Модель с заранее выделенными векторами цвета

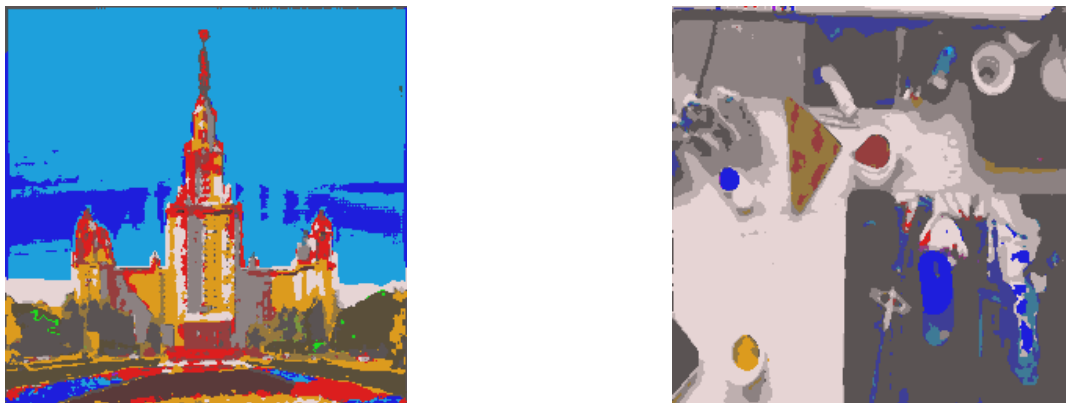


Рис. 7 Пример работы модели

В случае сегментации на области постоянного цвета модель местами странно определяет области постоянного цвета к примеру небо на фото слева. На простых сценах справляется визуально хорошо.

### 3.1.3 Суперпиксели



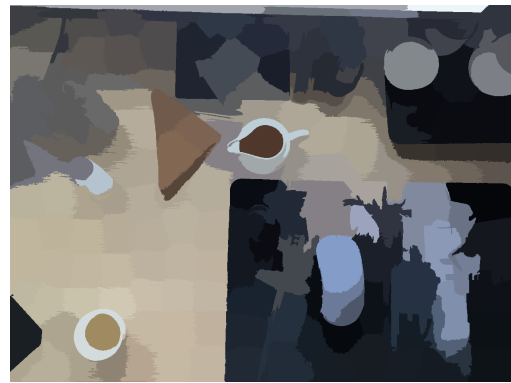


Рис. 8 Пример работы метода

Как видно метод теряет границы объектов и суперпиксели не являются непрерывной структурой, каждый суперпиксель даже с похожим цветом не объединяется в больший кластер с соседним.

### 3.1.4 K-means

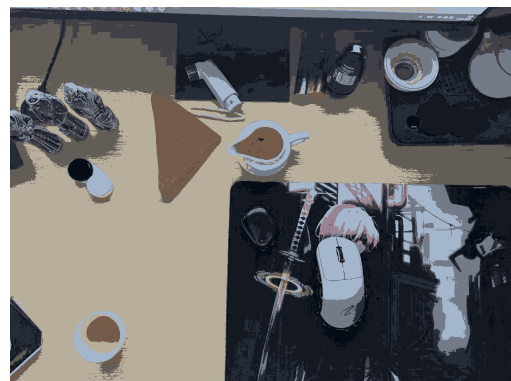


Рис. 9 пример работы алгоритма

Применим метод поиска отличия по форме для результатов сегментации на области постоянных цветов. Затем найдем отличия по форме между изображениями.



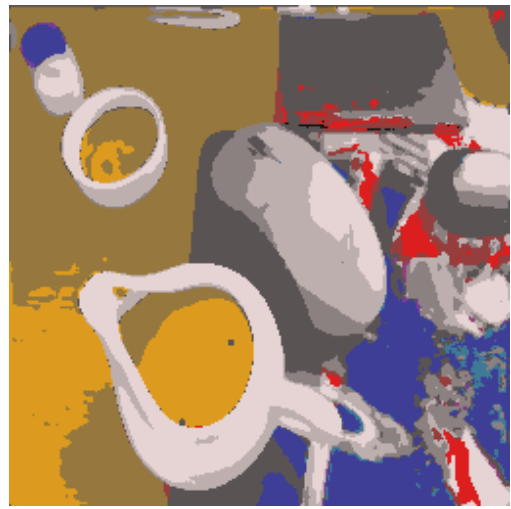
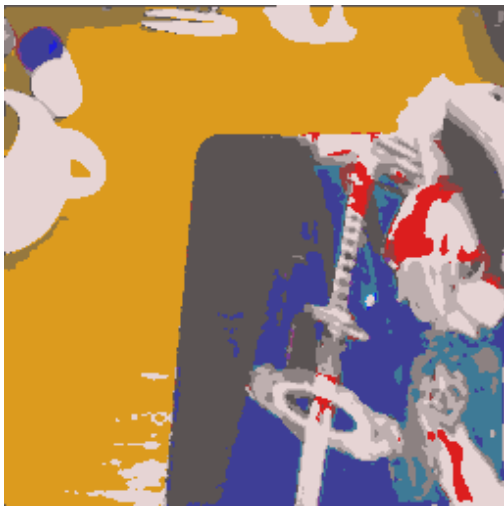


Рис. 10 результат использования метода заранее выделенных векторов цвета

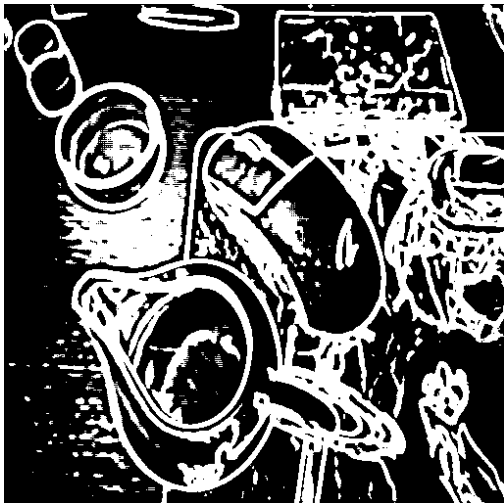


Рис. 11 результат применения модели обученной на поиск границ

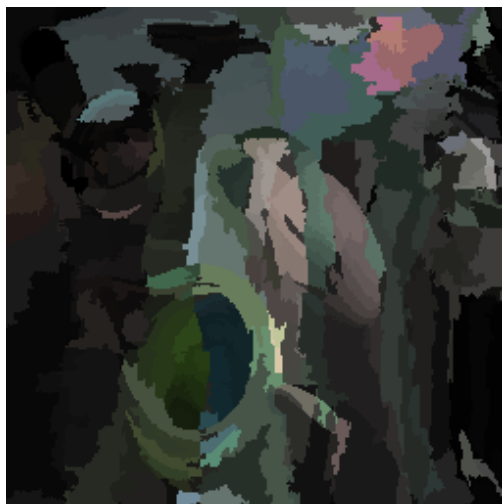
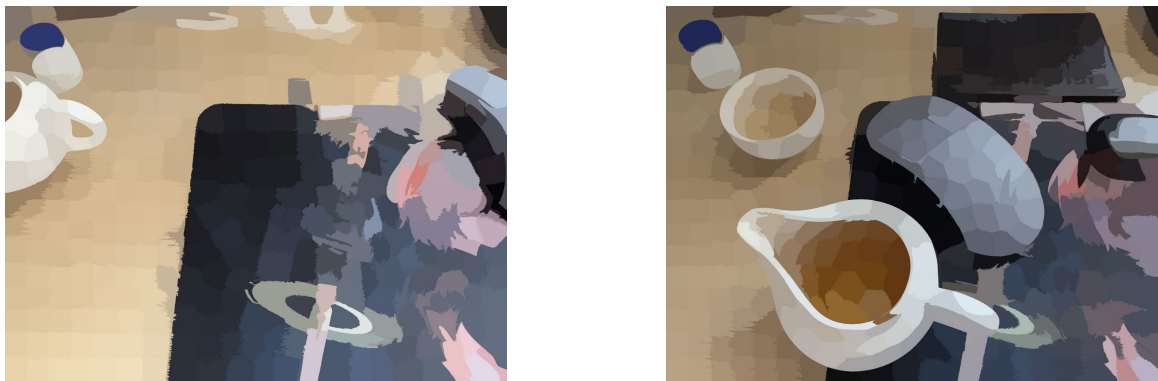


Рис. 12 Результат применения метода сегментации суперпикселями



Рис. 13 Результат вычитания при использовании K-средних

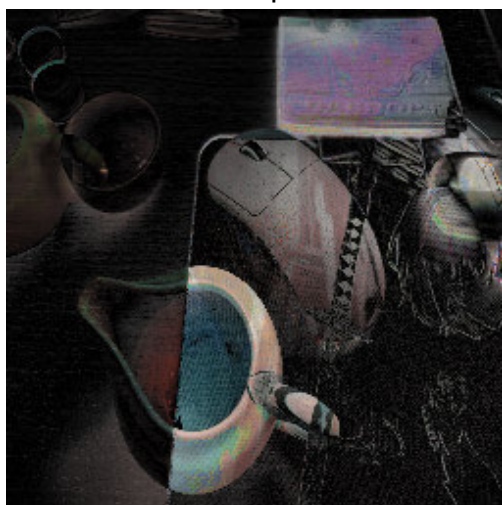


Рис. 14 Результат простого вычитания пикселей

Из результатов поиска отличий видно, что модель, с заранее выделенными векторами, хорошо справляется с поставленной задачей и самое главное кроме цветовой сохраняет еще и сегментационную информативность. К-средних имеет много шумов, но неплохо справляется.

### 3.3 Анализ результата

Для получения численной оценки ошибки методов, сравним результат поиска отличий по форме между изображениями сегментированными на области постоянного цвета человеком и между сегментированными методами из этой работы.

#### 3.3.1 Валидация Методов с помощью простых сцен

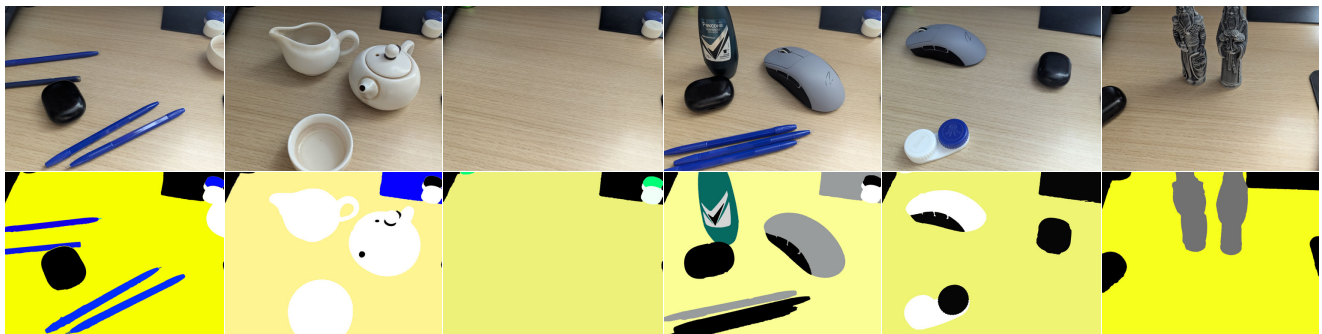


Рис. 15 Пример изображений по которым мы будем подбирать наилучшую модель

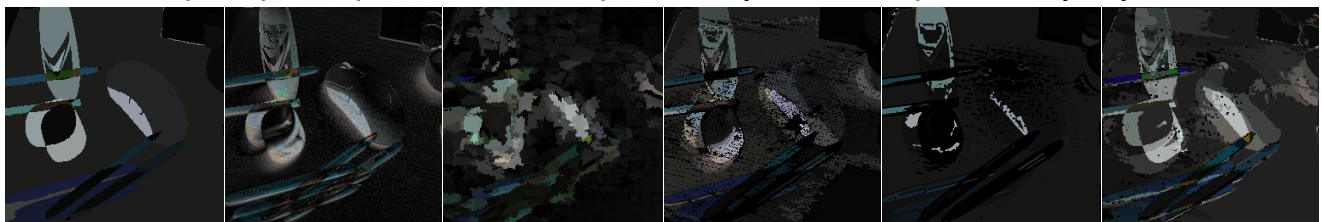


Рис. 16 Результаты поиска отличий между сценами 1 и 3. Методы(слева направо): Человеческая сегментация, по пикселям, суперпиксели, к-средних, модель обученная на границу, модель с заранее заданными цветами

Все отличия были нормированы и расстояние между двумя точками представляет из себя :

$$\Delta f(x) = \sum_{j=0}^n \left| \frac{f1_j(x) - f2_j(x)}{f1_j(x) + f2_j(x) + (5)} \right|$$

После сравнения в получились такие ошибки по 15 изображениям отличий между сценами.

	MAE( средняя абсолютная ошибка)	MSE(среднеквадратичная ошибка)	другие нормы по мере необходимости
модель с заранее выделенными	0.28860984974165843	0.16445787960674912	

	MAE( средняя абсолютная ошибка)	MSE(среднеквадратичная ошибка)	другие нормы по мере необходимос
векторами цвета			
модель обученная на границу	0.443609417292308577	0.33157599016087536	
K-means	0.36043508185199824	0.2439830254959446	
суперпиксели	0.43854002189599534	0.30205635620400073	
простое вычитание	0.4076589444582277	0.27521226817615213	

## 4 Выводы

Из результатов отклонений видно что лучше всего справилась модель с заранее заданными цветами.

### Эффективность методов:

- . Ошибка разметки самих изображений человеческим зрением как можно заметить блики и различные тени или отличия по яркости и контрасту были плохо размечены
- . Наилучшие результаты по метрикам MAE (0.288) и MSE (0.164) продемонстрировала модель, обученная на выделение областей постоянного цвета. Это связано с её способностью учитывать цветовые паттерны и структурные особенности изображений, что минимизирует ошибки при сравнении форм.
- . Метод суперпикселей (SLIC) и простое вычитание пикселей показали более высокие значения ошибок (MAE: 0.438–0.407, MSE: 0.302–0.275), что обусловлено их чувствительностью к шуму и отсутствием семантической адаптации.
- . Плохой результат модели обученной на границу можно аргументировать тем, что модель имела не достаточной хороший синтетический датасет, и из-за толщины границы(по обучающей выборке примерно 3 пикселя, что много для изображения 256x256 и так же дает вклад в ошибку), а так же в целом видно, что не хватает реальных изображений или изображений с текстурами в датасете, для увеличения обобщающей способности

## 5 Дальнейшие перспективы

### 1. Улучшение датасетов:

- Интеграция реальных изображений в синтетический датасет для повышения обобщающей способности моделей.
- Генерация данных с вариациями освещения, текстуры и перспективы для обучения устойчивых моделей.

### 2. Модернизация архитектур:



- Замена U-Net на более сложные и современные архитектуры (например U-Net++ или другие) для лучшего результата
- У лучшей модели(заранее заданные вектора) можно попробовать подобрать другое оптимальное количество векторов


### 3. Улучшение реализации:

- Можно было разбить изображение на кусочки и применять модель на них вместо сжатия изображения до стандартного размера, а также обучить модель на входное изображение размером 512x512, что так же бы улучшило качество.

### 4. Применение к видео:

- Попробовать применить модель и метод к видео с неподвижной камерой и как либо применить его для анализа.

## Приложения:

1. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv preprint arXiv:1505.04597*.
  - DOI: [10.48550/arXiv.1505.04597](https://doi.org/10.48550/arXiv.1505.04597)
  - PDF: [arxiv.org/pdf/1505.04597.pdf](https://arxiv.org/pdf/1505.04597.pdf)
2. Goodfellow I., Bengio Y., Courville A. (2016). *Deep Learning*. MIT Press. 800 p.
3. Machine Learning Collection by Aladdin Persson  
 [GitHub Stars](#)
4. **Пытьев Ю.П., Чуличков А.И.** Методы морфологического анализа изображений. — М.: Физматлит, 2010. — 336 с. **ISBN:** 978-5-9221-1210-7
5. Ссылка на мой репозиторий с частью кода  
 ([https://github.com/pechenie8v8kletku/Course\\_work\\_phys\\_phac/tree/main](https://github.com/pechenie8v8kletku/Course_work_phys_phac/tree/main))