САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Митрофанов Егор Владимирович

Выпускная квалификационная работа

Сегментация объектов интереса в видеопотоке

Уровень образования: бакалавриат

Направление 02.03.02 “Фундаментальные информатика и информационные технологии”

Основная образовательная программа “Программирование и информационные технологии”

Профиль “Автоматизация научных исследований”

Научный руководитель:

кандидат технических наук,

доцент,

Гришкин В.М.

Рецензент:

Доктор физико-математических наук,

профессор кафедры информационных систем,

Матросов А.В.

Санкт-Петербург

2020

Оглавление

Оглавление 2
Введение 3
Постановка задачи 5
1. Формальное описание 5
2. Работа с данными 5
3. Оценка результатов 5
4. Основные этапы работы 6
Обзор литературы 7
Глава 1: Методы сегментации изображений и видео 10
1.1 Метод MSER 10
1.2 Метод выделения связных компонент 10
1.3 Графо-ориентированная сегментация 12
1.4 Метод водораздела 12
1.5 Метод глубокого обучения 14
Глава 2: Реализация и тестирование системы 18
2.1 Основная суть алгоритма 18
2.2 Выбор программных средств 18
2.3 Реализация генератора данных 19
2.4 Реализация системы отделения фона 21
2.5 Реализация сегментации объектов 22
2.6 Исследование системы 23
Заключение 26
Список литературы 27
Приложение 29

Введение

В связи со стремительным развитием мощностей вычислительной техники обработка многомерных данных большого объема стала осуществима фактически в реальном времени. Это привело к массовой автоматизации и созданию систем, способных без вмешательства человека проводить анализ и выполнять сложные действия. В последнее время популярностью пользуются автоматические системы распознавания образов, которые позволяют автоматизировать многие трудоёмкие для человека процессы. Такие системы основаны на различных алгоритмах сегментации.

Особенную сложность при сегментации представляют изображения, содержащие скопления однотипных (однородных) объектов. Большинство алгоритмов определит всю группу объектов в один большой кластер, и если данные объекты перекрываются друг с другом, то невозможно будет сказать один это объект или несколько.

Под однотипными объектами понимаются объекты имеющие одинаковую природу (состав, цвет, прочность, форму), но в некоторой степени различающиеся по контуру и размеру. Примеры таких объектов приведены на рис 1.1.

Рис. 1 примеры однотипных (однородных) объектов

Данная задача находит применение во многих сферах:

- Медицина – сегментация клеток и метастазов на медицинских снимках (обычно имеющих огромное разрешение)
- География – сегментация крыш домов на снимках со спутника
- Промышленность – компьютерное зрение для роботов, выполняющих производственные задачи
- И многое другое

В данной работе предлагается идея алгоритма компьютерного зрения для определения среди однотипных объектов экземпляров наибольшего размера в видеопотоке, а также проведено его тестирование на сгенерированном дата сете. Такой алгоритм может применяться в реальных системах производственных предприятий и был бы полезен, например, для автоматизированного отделения больших минералов от маленьких на конвейере.

Постановка задачи

1. Формальное описание

Разработать алгоритм, который покадрово получает на вход видеопоток с размерностью кадров X × Y × N, где X и Y - кол-во пикселей по ширине и высоте, N - количество цветовых каналов. И для каждого полученного на вход кадра, выдает данные о расположении обнаруженных сегментов в виде матрицы X × Y, где каждая ячейка имеет одно целочисленное значение – номер класса к которому принадлежит данный пиксель изображения.

2. Работа с данными

В связи с отсутствием открытых баз данных с реальных предприятий была проведена генерация дата сета видеофрагментов с объектами на движущемся конвейере. Более подробно алгоритм генерации описан в главе 2.

3. Оценка результатов

Для облегчения процесса тестирования было решено разделить его на две части:

a) ***Степень точности отделения больших экземпляров от малых.***

Данный показатель определяет на сколько точно система отделяет однотипные объекты большего размера от малых(фона). Для оценки данного показателя был выбрана стандартная метрика IoU - (Intersection of Union):

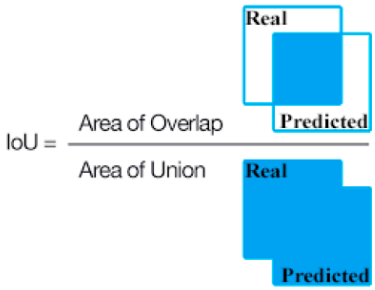

Рис. 2 метрика IoU

b) ***Количество объектов, находящихся на кадре в данный момент.***

Данный показатель считается как отношение количества классов, которые определила система (кроме фона), к реальному числу объектов на кадре.

4. Основные этапы работы

1. Провести анализ существующих методов сегментации объектов в видеопотоке

2. На основе анализа этих методов выбрать наиболее подходящие для использования при решении поставленной задачи сегментации.

3. Разработать метод сегментации однородных объектов в видеопотоке.

4. Реализовать систему для сегментации однородных объектов в видеопотоке

5. Провести экспериментальное исследование системы.

Обзор литературы

Перед разработкой системы было проанализировано некоторое количество научных публикаций по данной теме за последние 5 лет. Большинство алгоритмов для похожих задач специализированы для медицины.

В публикации [1] рассматривается подход глубокого обучения для сегментации медицинских снимков для различных существующих архитектур. Наивысшую точность по большинству метрик показывает архитектура U-net.

Данные обучения являются ключевым компонентом подходов к глубокому обучению, но их трудно получить для специализированных областей, часто встречающихся в робототехнике [2]. В данной статье предлагается идея генерации датасета для обучения, размещая объекты на изображении и на маске соответственно.

Анализы клеточной функции, особенно с использованием флуоресцентных пятен, повсеместно используются в биологических и медицинских науках. Несмотря на достижения в области компьютерного зрения, такие изображения часто анализируются с использованием только ручных или элементарных автоматических процессов. Сегментация на основе водораздела является эффективным методом идентификации объектов на изображениях; он превосходит обычно используемые методы анализа изображений, но требует успешного ознакомления с методами компьютерного зрения. В статье [3] представляется и реализуется алгоритм анализа и классификации изображений на основе водораздела в графическом интерфейсе, что позволяет широкому кругу пользователей легко понять алгоритм и настроить параметры в соответствии с их конкретными потребностями.

В статье [4] предложен метод, состоящий из сегментации и этапа отслеживания, который включает исправление ошибок сегментации (отслеживание методом обнаружения). Для сегментации в качестве входных данных для последующей обработки водораздела используются прогнозы карт расстояний на основе глубокого обучения между сотами и новых карт расстояний соседей. Поскольку большинство предоставленных данных являются двумерными, двумерная сверточная нейронная сеть обучена предсказывать карты расстояний. Отслеживание основано на оценке движения в сочетании с согласованием, сформулированным как проблема минимальной стоимости максимального потока.

В процессе распознавания объектов все важные области, содержащие интересующие объекты, ограничены, а фон игнорируется. Обычно объект ограничивается прямоугольником, который выражается через пространственные координаты его верхнего левого угла, его ширины и высоты. Недостаток данного подхода состоит в том, что для объектов сложной формы ограничивающий прямоугольник также включает фон, который может занимать значительную часть области, поскольку ограничивающий прямоугольник не охватывает объект плотно. Такое поведение может снизить производительность классификатора, примененного к ограничивающей рамке, или может не соответствовать требованиям точного обнаружения. Для этого предлагается для каждого обнаруженного прямоугольника отделить объект, находящийся в нем, от фона. Проблема, на которой фокусируется публикация [5], заключается в создании точного детектора с сегментированием экземпляров и возможностью обработки в реальном времени на графических картах среднего уровня.

В публикации [6] предлагается специальный метод для обнаружения и идентификации воздушных судов, объединяющий две совершенно разные сверточные нейронные сети (CNN): модель сегментации, основанную на

модифицированной архитектуре U-net, и модель обнаружения, основанную на архитектуре RetinaNet. Результаты показывают, что эта комбинация значительно превосходит каждую унитарную модель, резко снижая уровень ложноотрицательных результатов.

Вывод:

Метод глубокого обучения широко применяется для сегментации однотипных объектов на медицинских снимках. При этом лучшую точность показывает архитектура U-net. Еще одним популярным алгоритмом является метод водораздела, который имеет множество вариаций и высокую скорость работы. Для областей где невозможно или трудно получить экспериментальные данные можно провести автоматическую генерацию датасета, что дает возможность как обучать модели, так и тестировать их.

Глава 1: Методы сегментации изображений и видео

1.1 Метод MSER

Метод MSER (maximally stable extremal regions) основан на выделении на изображении областей, обладающих следующими свойствами:

a) множество замкнуто при непрерывном преобразовании координат изображения
b) множество замкнуто при монотонном преобразовании интенсивности изображения.

Если интенсивность пикселя больше порога он считается белым, если меньше – черным. Таким образом, определяется список изображений от белого цвета к черному, с помощью которого можно построить множество связных компонент интенсивности.

Данный метод позволяет эффективно (алгоритм имеет практически линейное время работы) отслеживать определенные объекты в видеопотоке при изменении ракурса камеры, однако плохо справляется с разделением однотипных объектов [7].

1.2 Метод выделения связных компонент

Данный способ основан на методах обхода графов. Как только первый пиксель подключенного компонента найден, все соседние пиксели этого компонента помечаются перед переходом к следующему пикселю в изображении. Для этого формируется связанный список, в котором будут храниться индексы пикселей, которые связаны друг с другом. Метод

построения связанного списка определяет использование поиска по глубине или ширине. Для этого метода нет разницы, какую стратегию использовать.

Предполагается, что входное изображение является двоичным изображением с пикселями, являющимися либо фоном, либо передним планом, и требуются найти связные компоненты в пикселях переднего плана. Алгоритм:

1) Начать с первого пикселя на изображении. Установить текущую метку на 1. Перейти к (2).
2) Если этот пиксель является пикселем переднего плана, и он еще не помечен, присвоить ему текущую метку и добавить его в качестве первого элемента в очереди, затем перейти к (3). Если это фоновый пиксель или он уже помечен, повторить (2) для следующего пикселя изображения.
3) Извлечь элемент из очереди и посмотреть на его соседей (в зависимости от типа подключения). Если сосед является пикселем переднего плана и еще не помечен, присвоить ему текущую метку и добавить его в очередь. Повторять (3) до тех пор, пока в очереди больше не будет элементов.
4) Перейти к (2) для следующего пикселя в изображении и увеличить текущую метку на 1.

Пиксели помечаются перед помещением в очередь, таким образом в очереди будет только пиксель для проверки соседей и добавления их в очередь, если это необходимо. Этот алгоритм должен проверять соседей каждого пикселя переднего плана один раз и не проверяет соседей пикселей фона [8].

1.3 Графо-ориентированная сегментация

В данном подходе изображения представляются в виде графов, где вершинами являются пиксели, а ребрами мера различия соседних пикселей. Данная мера выбирается в зависимости от целей сегментации: это может быть разность в яркости или цвете, положение относительно других пикселей или некоторое другое локальное значение.

Разбиение на классы происходит, как разделение графа на подграфы в соответствии с заданными порогами соответствующих значений.

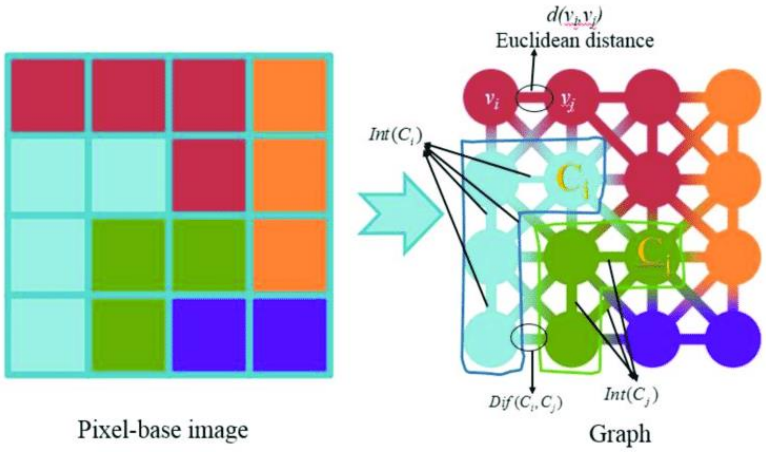

Рис 1.1 Иллюстрация графо-ориентированного подхода

Недостатком данного метода является сложность подбора параметров для конкретной задачи и низкая точность системы при плохой их настройке.

1.4 Метод водораздела

Преобразование водораздела обрабатывает изображение, как топографическую карту с значением каждой точки, представляющей ее высоту, и находит линии, проходящие вдоль вершин гребней. Есть много разных алгоритмов для вычисления водоразделов:

а) Водораздел наводнением

Метод был предложен в 1979 году С. Бехером и К. Лантюжулом. Основная идея состоит в том, чтобы поместить источник воды в каждом региональном минимуме на изображении, затопить весь рельеф из источников и построить барьеры, когда встречаются разные источники воды. Результирующий набор барьеров представляет собой водораздел наводнением [9].

b) Водораздел по топографической дистанции

Интуитивно понятно, что капля воды, падающая на рельеф, течет к «ближайшему» минимуму. «Ближайший» минимум - это тот минимум, который лежит в конце пути наискорейшего спуска. С точки зрения топографии это происходит, если точка лежит в водосборном бассейне этого минимума. Так для каждой точки определяется класс, к которому она принадлежит.

c) Межпиксельный водораздел

Алгоритм межпиксельной реализации метода водораздела [10]:

1) Маркировать каждый минимум отдельной меткой. Инициализировать набор S с помеченными узлами.
2) Извлечь из S узел x минимальной высоты. Присвоить метку x каждому немаркированному узлу y, смежному с x, и вставить y в S.
3) Повторять шаг 2, пока S не станет пустым.

Метод водораздела, в отличие от предыдущих методов позволяет сегментировать изображения даже с большим количеством пересекающихся однородных объектов, и так же имеет линейное время работы. Сложность заключается в том, что данный метод очень чувствителен к помехам на изображении, поэтому необходима его существенная предобработка.

1.5 Метод глубокого обучения

Существует большое количество архитектур нейронных сетей, специализированных под различные цели. Самая простая архитектура – персептрон позволяет выполнять классификацию объектов. Рекуррентные нейронные сети показывают хорошие результаты при работе с последовательностями.

В наши дни наилучшие результаты для сегментации показывают сверточные нейронные сети (CNN). Их принцип работы основан на постепенном уменьшении размерности исходного изображения, за счет расширения размерности карт признаков. Таким образом на последнем сверточном слое обычно получается вектор признаков изображения. Затем данный вектор передается обычному классификатору или же еще одной нейросети.

Архитектура CNN:

1) Слой свертки

Данный блок содержит для каждого выходного канала свой фильтр (ядро свертки). Данный фильтр по фрагментам обрабатывает предыдущий слой (веса фильтров находятся в процессе обучения). На выходе каждый слой дает карты признаков для каждого фильтра и передает их следующему блоку.

2) Слой активации

Результат каждой свертки передается в функцию активации (нелинейная функция определяющая выходной сигнал нейрона). В персептроне обычно используется функция гиперболического тангенса или сигмоида, однако для глубоких сетей была найдена более простая и эффективная функция ReLU (rectified linear unit), которая представляет собой отсечение отрицательной части скалярной величины.

3) Слой пулинга

На данном слое происходит уменьшение размерности карты признаков путем замены блоков пикселей (обычно 2x2) одним. Данный слой позволяет сети не переобучаться и не хранить большое количество ненужных данных и обычно ставится после каждого слоя свертки.

4) Полносвязная нейронная сеть

После всех слоев свертки обычно остается карта признаков, где все признаки представлены скалярными векторами малой размерности. Данные вектора объединяются и передаются полносвязной нейронной сети (возможно многослойной), которая уже выполняет классификацию или сегментацию.

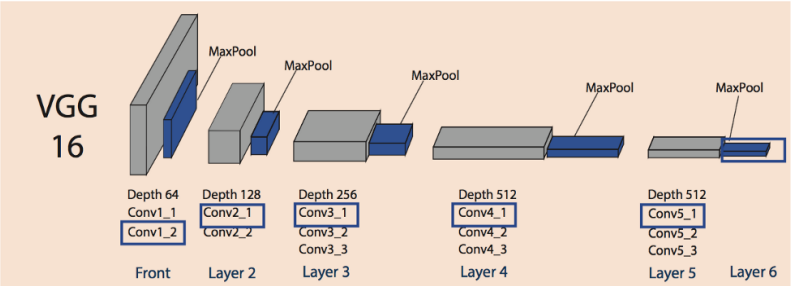

Рис. 1.2 Пример архитектуры CNN

Архитектура U-net:

Информация, получаемая на каждом слое свертки, является сжатой картой признаков соответствующей размерности. Это свойство и использует данная архитектура. На каждом блоке расширения размерности текущая карта признаков объединяется с соответствующей картой признаков сверточного слоя и предается следующему блоку. Таким образом, очертания объектов получаются более четкими и гладкими, и требуется меньше данных для обучения.

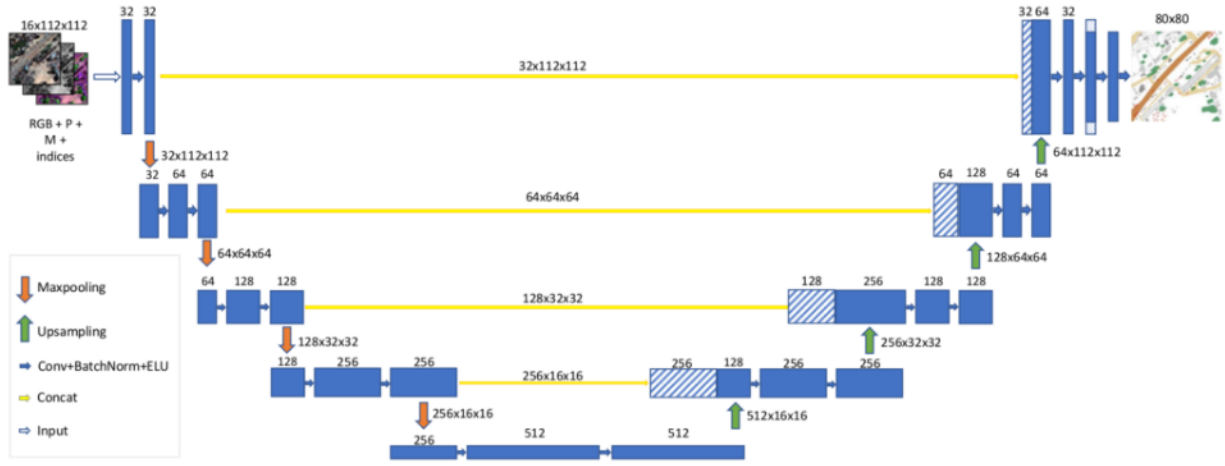

Рис. 1.3 Пример архитектуры U-net

Архитектура сети показана на рисунке 1.3. Она состоит из сжимающего пути (левая сторона) и расширяющего пути (правая сторона). Архитектура блоков следует типичной архитектуре сверточной сети. Они состоят из применения двух сверток 3x3, за которыми следует активация ReLU и операции объединения пула 2x2 (выбор максимума) с шагом 2 для понижающей дискретизации. На каждом шаге понижающей дискретизации удваивается количество каналов. Каждый шаг в расширяющем пути состоит из повышения дискретизации карты объектов: свертка 2x2 («свертка вверх»), которая вдвое сокращает число каналов, конкатенации с соответствующим образом обрезанной картой объектов из сжимающего пути и двух 3x3 сверток, за каждой из которых следует ReLU активация. Обрезка необходима из-за потери граничных пикселей в каждой свертке. На последнем уровне свертка 1x1 используется для отображения каждого 64-компонентного вектора признаков на требуемое количество классов. Всего в оригинальной сети 23 сверточных слоя [11].

Достоинствами метода глубокого обучения является высокая точность сегментации, устойчивость изображения к поворотам и сдвигам, возможность распараллеливания вычислений. Основные недостатки –

сложность выбора параметров и архитектуры сети для конкретной задачи, необходимость наличия обучающей выборки, большое время работы для сетей со сложной структурой.

Вывод:

По проанализированным данным была построена таблица (таблица 1.1), где каждому свойству была проставлена оценка от 1 до 5, где 1 означает, что данный показатель является достоинством (самое низкое время работы, самая высокая точность) и соответственно 5 – наоборот.

Особенности метода	MSER	Метод связных компонент	Графо-ориентированный метод	Метод водораздела	Метод глубокого обучения
Время работы	1	1	1	1	3
Время на подготовку	1	1	1	1	3
Предобработка данных	2	1	1	4	3
Точность метода	3	4	3	2	1
Применимость к сегментации однородных объектов	4	5	3	1	1

Таблица 1.1 Сравнение методов сегментации

Метод глубокого обучения дает большую точность, однако сложные сети работают слишком долго. Метод водораздела выделяется своей скоростью работы, но с достаточно низкой точностью при стандартной предобработке изображений.

Поэтому предлагается идея объединения достоинств двух данных методов для решения поставленной задачи. Структура сети должна быть небольшой и компактной и выполнять простую задачу отделения требуемых объектов от фона, а метод водораздела работать с полученными данными.

Глава 2: Реализация и тестирование системы

2.1 Основная суть алгоритма

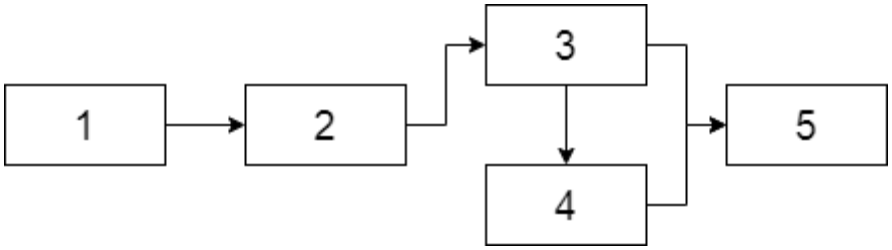

Рис. 2.1 схема алгоритма

1) Система принимает на вход очередной кадр (возможно в режиме реального времени)
2) С помощью обученного классификатора большие объекты отделяются от фона (маленьких объектов)
3) На полученной бинарной маске находится преобразование distance transform
4) Определяются маркеры объектов, как локальные максимумы на distance transform матрице
5) На бинарной маске применяется преобразование водораздела по полученным маркерам

2.2 Выбор программных средств

Язык Python содержит большое количество библиотек для работы с нейронными сетями. Поэтому для разработки системы был выбрана самая стабильная версия 3.7.6, поддерживающая большее количество библиотек.

Одной из самых используемых и известных библиотек для машинного обучения является TensorFlow, также существует множество надстроек и вспомогательных библиотек. Одной из таких надстроек является библиотека

keras, который позволяет удобно и компактно описывать архитектуру нейросети, не вдаваясь в описывание отдельных нейронов, а работая со слоями. С помощью данной библиотеки можно строить сети различных архитектур, настраивать параметры обучения, а также получать указанные метрики на каждом шаге обучения.

Для работы с видео и изображениями была выбрана стандартная библиотека opencv. Она позволяет покадрово работать с видеофрагментами даже в режиме реального времени, а также записывать результат в видеопоток. Картинки представляются в виде матрицы соответствующей размерности, которую и обрабатывает система.

Работа с векторами и матрицами происходила с помощью библиотеки numpy, которая имеет большое количество встроенных матричных операций, что позволяет удобно работать со считанными изображениями. Например, данная библиотека позволяет делать срезы многомерных массивов по каждой размерности, а также производить логические операции над всеми элементами матрицы сразу.

Преобразование водораздела и поиск локальных максимумов в последней части системы производится с помощью библиотеки skimage, так как в ней данные функции реализованы оптимально и имеют возможность настройки параметров под конкретную задачу.

2.3 Реализация генератора данных

Для обучения классификатора и тестирования системы было решено моделировать движение эллиптических объектов с цветами градаций серого на конвейере. Для этого на матрицу размером 256 x 1256 x 3 с центрами в случайных координатах, наносились эллипсы, радиусы которых тоже были заданы случайно. При этом на бинарную матрицу размера 256 x 1256

наносились только эллипсы большего размера (которые требуется классифицировать).

Так как в силу диффузии меньшие объекты под действием тряски (которая присутствует на конвейере) опускаются ниже, а нижние объекты находятся под тенью верхних, для формирования фона были созданы 1000 эллипсов оба радиуса которых являются случайной целой величиной от 10 до 20, а их цвет задан случайным числом от 0 до 127 (более темные градации серого).

Далее случайным образом было размещено 20 объектов, которые представляют собой эллипсы, оба радиуса которых заданы случайным целым числом от 30 до 50, а цвет от 128 до 255.

Полученная матрица была разбита на видеофайл, последовательным проходом окном 256 x 256. Пример исходной матрицы, маски и отдельного кадра на рис. 2.2-2.4.

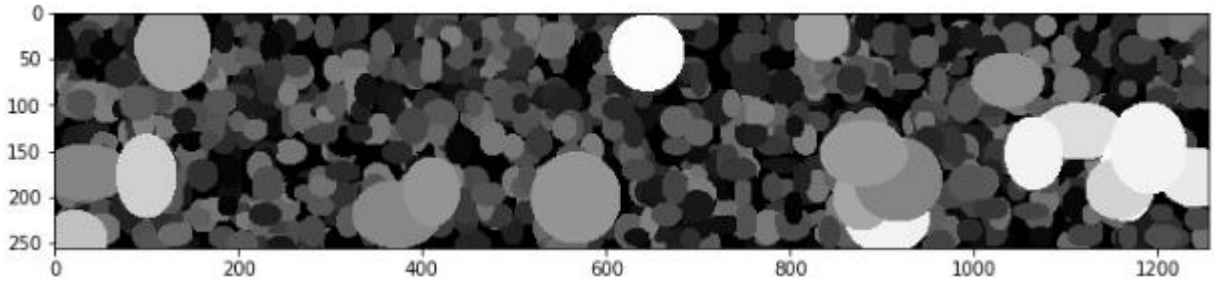

Рис. 2.2 Исходная матрица

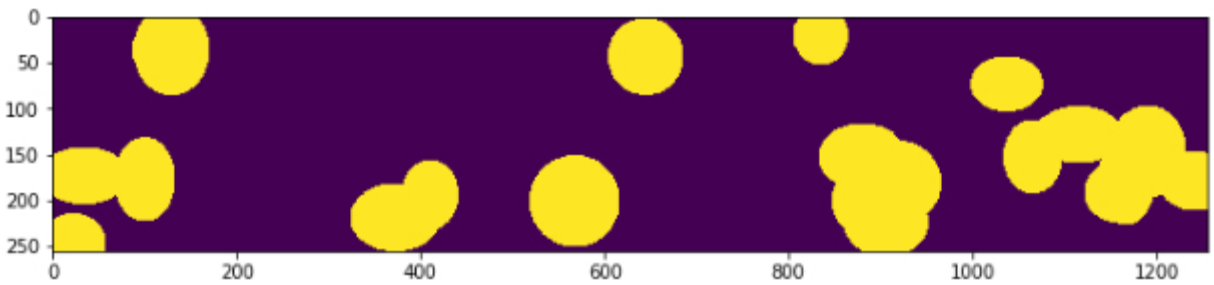

Рис. 2.3 Маска исходной матрицы

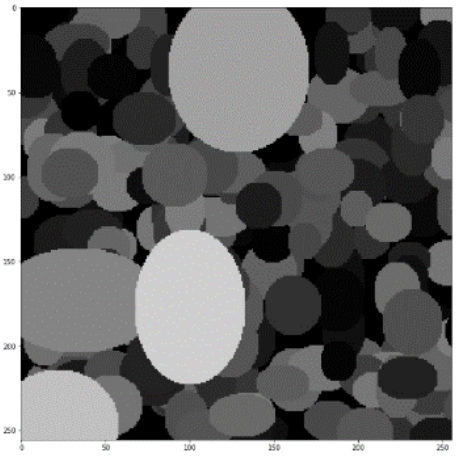

Рис. 2.4 Отдельный кадр полученного видеофрагмента

2.4 Реализация системы отделения фона

В последнее время в задаче сегментации лучше всего показывают себя сверточные нейронные сети (CNN). Существует много библиотек, позволяющих использовать готовые нейросети, содержащие большое количество слоев и обученные на миллионах изображений. Однако в данной задаче одним из факторов является скорость работы системы, а чем сложнее структура сети, тем дольше происходит предсказание ответа. Поэтому было решено разделить исходную задачу: определение контуров необходимых объектов реализовать с помощью подхода глубокого обучения, а разделение объектов на классы с помощью метода watershed.

С помощью фреймворка Keras была построена 4х слойная нейронная сеть по архитектуре U-net. На вход подается картинка в виде numpy массива, а на выходе получается матрица, где для каждого пикселя исходного изображения содержится вероятность, что это требуемый объект. Далее происходит изменение матрицы на бинарную путем сравнения с пороговым значением, которое подбиралось эмпирически. Построенная сеть была обучена на сгенерированном датасете из 20000 изображений.

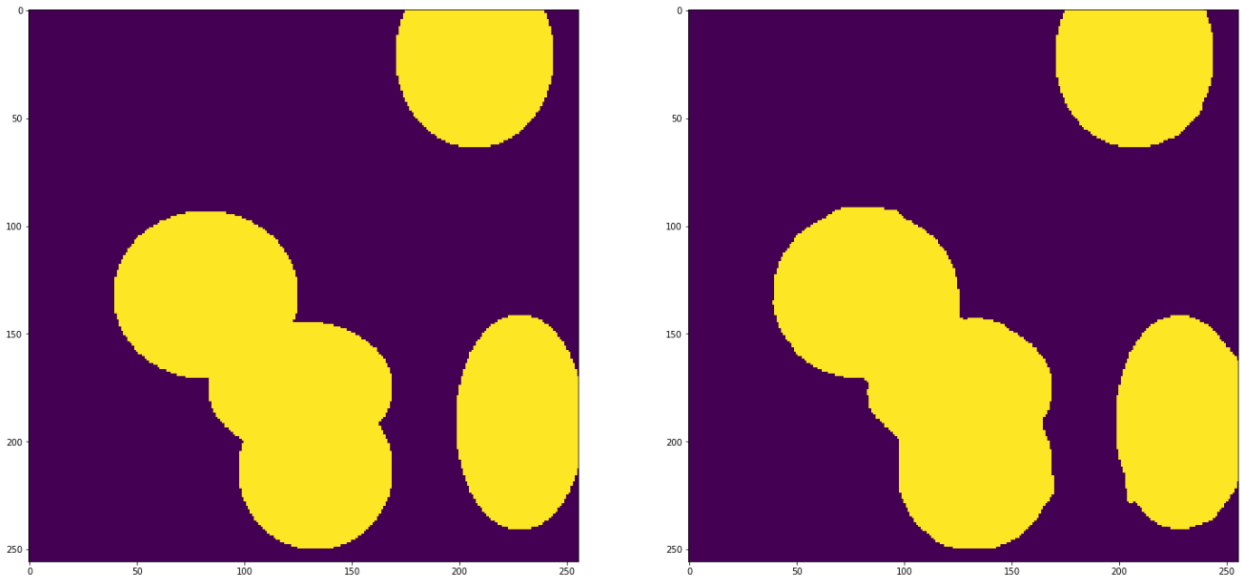

Рис. 2.5 Сравнение реальной маски(слева) с построенной нейросетью(справа)

2.5 Реализация сегментации объектов

Если сразу применять метод водораздела к исходному изображению, некоторые точки маленьких объектов будут попадать на локальные максимумы, что приведет к ошибке, поэтому отделение больших объектов от маленьких было выполнено с помощью метода глубокого обучения.

Так как однотипные объекты имеют примерно одинаковую форму, из всех вариантов был выбран метод водораздела на основе топографического расстояния.

На полученной бинарной маске находится преобразование distance transform, чем ближе пиксель к центру объекта, тем больше его значение (от 0 до 1). Определяются маркеры объектов, как локальные максимумы на distance transform матрице. На бинарной маске применяется преобразование водораздела по полученным маркерам.

Итоговая матрица имеет целочисленные значения от 0 до n - 1, где n это количество классов, при этом классу с индексом 0 всегда соответствует фон. До работы над видеофрагментом для каждого из классов генерируются

случайные цвета, и затем связываются с определенными классами. На первом шаге номеру класса соответствует номер его цвета.

Классы объектов на текущем кадре не обязательно соответствуют классам на следующем, так как могли появиться новые точки и порядок обработки изменился. Для этого сохраняется следующая информация:

- Матрица сегментации предыдущего кадра
- Цвета, которые соответствуют каждому классу

Каждый кадр, начиная со второго, сравнивается с предыдущим, и, если объект уже присутствовал ранее, то его классу присваивается соответствующий цвет.

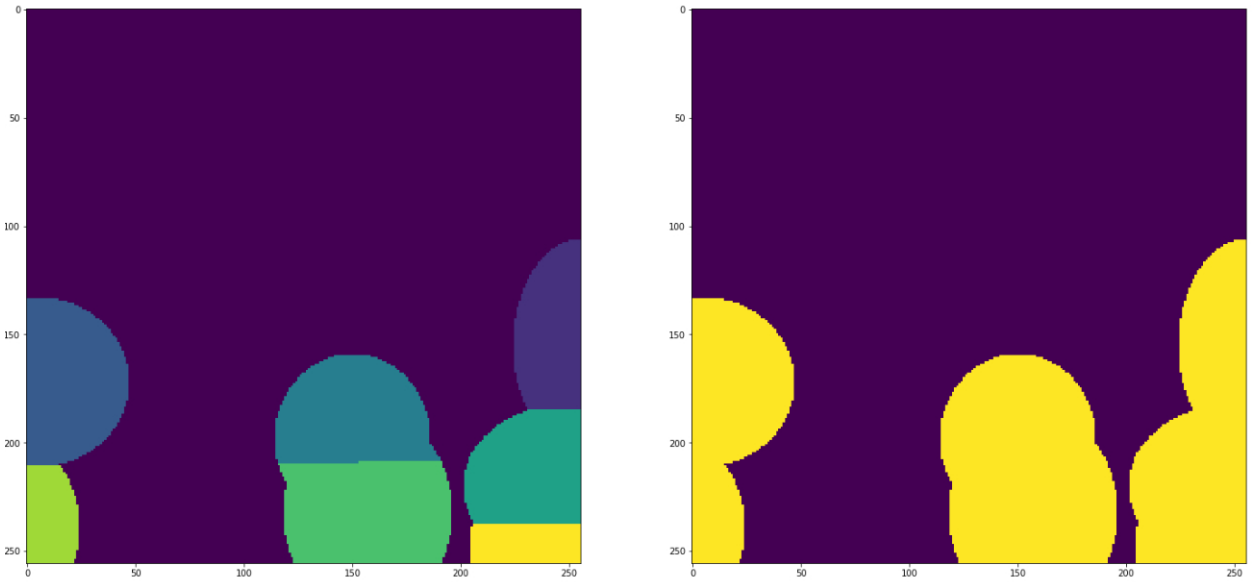

Рис. 2.6 слева результат работы на маске справа.

2.6 Исследование системы

Для исследования системы с помощью того же генератора были смоделированы пять видеофрагментов по 1000 кадров каждый. Полученные результаты занесены в таблицу 2.1.

Номер видеофрагмента	Наихудшая точность IoU	Наилучшая точность IoU	Средне кадровая точность IoU	Точность по числу классов
1	0.962	0.999	0.997	0.895
2	0.961	0.999	0.992	0.897
3	0.993	0.999	0.996	0.869
4	0.935	0.998	0.997	0.958
5	0.990	0.999	0.999	0.897

Таблица 2.1 Показатели точности системы

Номер видеофрагмента	Среднее время работы первой части сек./кадр	Среднее время работы второй части сек./кадр	Среднее время работы системы сек/кадр
1	0.369	0.046	0.415
2	0.362	0.046	0.408
3	0.363	0.047	0.410
4	0.367	0.043	0.410
5	0.355	0.046	0.402

Таблица 2.2 Время работы системы

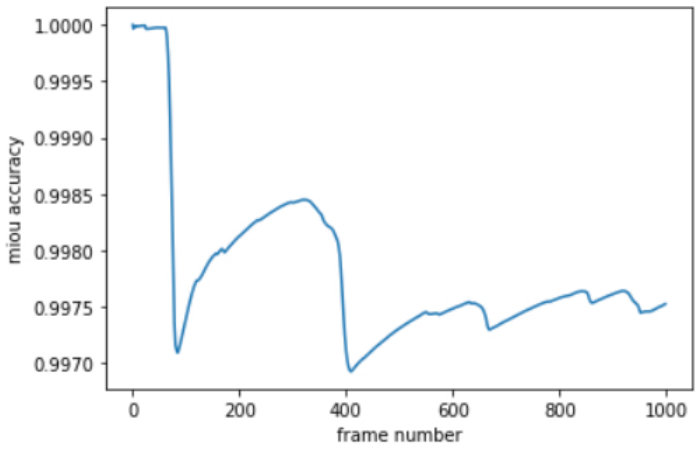

Рис. 2.7 Зависимость MIoU от числа обработанных кадров для первого видеофрагмента

Таблица 2.2 показывает время работы системы на соответствующих видеофрагментах для каждой части системы. Первая часть – отделение больших объектов от фона показывает наибольшую точность (около 99%), однако занимает большую часть времени работы системы по сравнению со второй частью – сегментация по классам.

Тестирование проводилось на ноутбуке с видеокартой NVIDIA GeForce GTX 950M. На обработку одного кадра системе требуется меньше

чем пол секунды. Исходя из этого, можно утверждать, что работа системы в реальном времени осуществима в промышленных масштабах.

Анализ полученных видеофрагментов показал, что большую часть ошибок система показывает для объектов, не полностью вошедших в кадр. Ошибок в сегментации объектов по центру практически нет. Таким образом, для робота, который выполняет сортировку, наиболее эффективно будет производить захват объектов по центру. Также если объекты имеют слишком большую площадь пересечения, то они определяются в один класс.

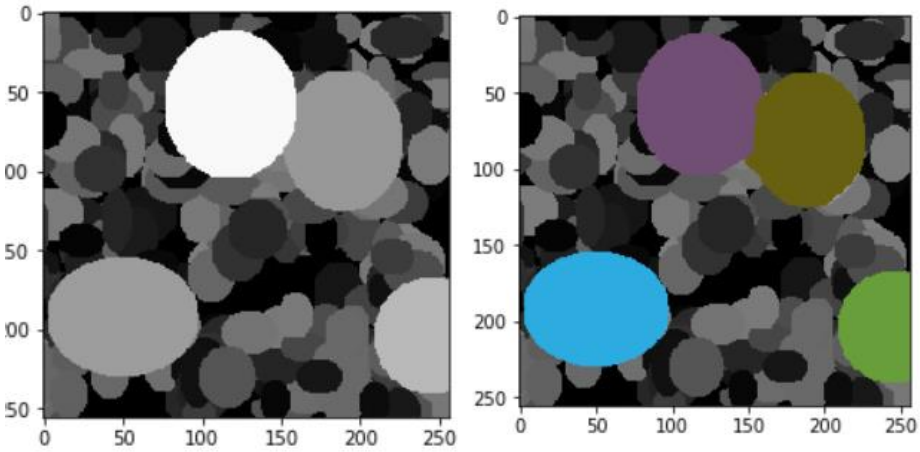

Рис. 2.8 Сравнения исходного кадра и результата работы системы на нем

Заключение

В рамках данной работы была поставлена задача: разработка системы сегментации движущихся однотипных объектов, выделяющихся по размеру. В ходе ее решения были поставлены подзадачи, описанные в пункте «основные этапы работы». В итоге были получены следующие результаты:

1) Проанализированы научные публикации по теме сегментации однородных объектов.
2) Проведен анализ методов сегментации.
3) Сгенерирован тестовый датасет для задачи отделения минералов крупного размера на конвейере.
4) Разработана система на основе проанализированных данных.
5) Проведено экспериментальное исследование системы.

Таким образом, можно утверждать, что поставленная задача решена полностью.

Стоит отметить, что в данной работе представлена реализация алгоритма под конкретную задачу, однако он может быть эффективен для более широкого круга проблем, и исследование, проведенное в данной работе, может быть продолжено.

Список литературы

[1] Beyond CNNs: Exploiting Further Inherent Symmetries in Medical Images for Segmentation. Shuchao Pang, Anan Du, Mehmet A. Orgun, Yan Wang, Quanzheng Sheng, Shoujin Wang, Xiaoshui Huang, Zhemei Yu

URL: https://arxiv.org/ftp/arxiv/papers/2005/2005.03924.pdf

[2] Stillleben: Realistic Scene Synthesis for Deep Learning in Robotics. Max Schwarz and Sven Behnke.

URL: https://arxiv.org/pdf/2005.05659.pdf

[3] A watershed-based algorithm to segment and classify cells in fluorescence microscopy images. Lena R. Bartell, Lawrence J. Bonassar, and Itai Cohen.

URL: https://arxiv.org/pdf/1706.00815.pdf

[4] Cell segmentation and tracking using distance transform predictions and movement estimation with graph-based matching. Tim Scherr, Katharina Löffler, Moritz Böhland, Ralf Mikut.

URL: https://arxiv.org/pdf/2004.01486.pdf

[5] POLY-YOLO: higher speed, more precise detection and instance segmentation for YOLOV3. 2020 Petr Hurtik, Vojtech Molek, Jan Hula, Marek Vajgl, Pavel Vlasanek , and Tomas Nejezchleba

URL: https://arxiv.org/pdf/2005.13243.pdf

[6] Concurrent segmentation and object detection CNNS for aircraft detection and identification in satellite images. Damien Grosgeorge, Maxime Arbelot, Alex Goupilleau, Tugdual Ceillier, Renaud Allioux

URL: https://arxiv.org/pdf/2005.13215.pdf

[7] Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. J. Matas, O. Chum, M. Urban, T. Pajdla

URL: http://cmp.felk.cvut.cz/~matas/papers/matas-bmvc02.pdf

[8] Abubaker, A; Qahwaji, R; Ipson, S; Saleh, M. One Scan Connected Component Labeling Technique. Signal Processing and Communications, 2007

[9] Serge Beucher and Christian Lantuéj workshop on image processing, real-time edge and motion detection (1979).

[10] Serge Beucher and Fernand Meyer. The morphological approach to segmentation: the watershed transformation. In Mathematical Morphology in Image Processing (1993).

[11] U-Net: Convolutional Networks for Biomedical Image Segmentation. Olaf Ronneberger, Philipp Fischer, and Thomas Brox

URL: https://arxiv.org/pdf/1505.04597.pdf

Приложение

Код генератора датасета:

```
for k in range(5):

    kol = np.zeros((1000))
    img = np.zeros((256,1256,3),  int)

    for i in range(1000):
        r = rnd.randint(0,255)
        c = rnd.randint(0, 1256)
        r_rad = rnd.randint(10,20)
        c_rad = rnd.randint(10,20)
        val = rnd.randint(0,128)
        color = [ val for k in range(3)]
        img[draw.ellipse(r,c,r_rad,c_rad, shape=(256,1256))]=color

    img2 = np.array(img)
    mask = np.zeros((256,1256),  bool)

    for i in range(20):
        r = rnd.randint(0,255)
        c = rnd.randint(0, 1256)
        r_rad = rnd.randint(30,50)
        c_rad = rnd.randint(30,50)
        #color = [rnd.randint(0,255) for k in range(3)]
        val = rnd.randint(128,256)
        color = [ val for k in range(3)]
        for ii in range(1000):
            if not ((ii > c - c_rad and ii > c + c_rad) or (ii + 255 < c - c_rad and ii + 255
< c + c_rad)):
                kol[ii] += 1
        img2[draw.ellipse(r,c,r_rad,c_rad, shape=(256,1256))]=color
        mask[draw.ellipse(r,c,r_rad,c_rad, shape=(256,1256))]=1

    img_path = 'data/gen1/test/images/'
    mask_path = 'data/gen1/test/masks/'
    kol_path = 'data/gen1/kol/'

    for l in range (1000):
        ll = k * 1000  + l
        img_mask = np.zeros((256,256,3),  int)
        img_mask[mask[:,l:l+256]>0]=[255,255,255]
```

```
        cv2.imwrite(img_path+str(ll)+'.jpg',img2[:,l:l+256])
        #print(img_path+str(l)+'.jpg')
        cv2.imwrite(mask_path+str(ll)+'.jpg',img_mask)
    np.save(kol_path+str(k)+'.npy', kol)
```

Построение и обучение нейросети:

```
inp = Input(shape=(256, 256, 3))

conv_1_1 = Conv2D(32, (3, 3), padding='same')(inp)
conv_1_1 = Activation('relu')(conv_1_1)

conv_1_2 = Conv2D(32, (3, 3), padding='same')(conv_1_1)
conv_1_2 = Activation('relu')(conv_1_2)

pool_1 = MaxPooling2D(2)(conv_1_2)


conv_2_1 = Conv2D(64, (3, 3), padding='same')(pool_1)
conv_2_1 = Activation('relu')(conv_2_1)

conv_2_2 = Conv2D(64, (3, 3), padding='same')(conv_2_1)
conv_2_2 = Activation('relu')(conv_2_2)

pool_2 = MaxPooling2D(2)(conv_2_2)


conv_3_1 = Conv2D(128, (3, 3), padding='same')(pool_2)
conv_3_1 = Activation('relu')(conv_3_1)

conv_3_2 = Conv2D(128, (3, 3), padding='same')(conv_3_1)
conv_3_2 = Activation('relu')(conv_3_2)

pool_3 = MaxPooling2D(2)(conv_3_2)


conv_4_1 = Conv2D(256, (3, 3), padding='same')(pool_3)
conv_4_1 = Activation('relu')(conv_4_1)

conv_4_2 = Conv2D(256, (3, 3), padding='same')(conv_4_1)
conv_4_2 = Activation('relu')(conv_4_2)

pool_4 = MaxPooling2D(2)(conv_4_2)
```

```
up_1 = UpSampling2D(2,  interpolation='bilinear')(pool_4)
conc_1 = Concatenate()([conv_4_2, up_1])

conv_up_1_1 = Conv2D(256, (3, 3), padding='same')(conc_1)
conv_up_1_1 = Activation('relu')(conv_up_1_1)

conv_up_1_2 = Conv2D(256, (3, 3), padding='same')(conv_up_1_1)
conv_up_1_2 = Activation('relu')(conv_up_1_2)


up_2 = UpSampling2D(2,  interpolation='bilinear')(conv_up_1_2)
conc_2 = Concatenate()([conv_3_2, up_2])

conv_up_2_1 = Conv2D(128, (3, 3), padding='same')(conc_2)
conv_up_2_1 = Activation('relu')(conv_up_2_1)

conv_up_2_2 = Conv2D(128, (3, 3), padding='same')(conv_up_2_1)
conv_up_2_2 = Activation('relu')(conv_up_2_2)


up_3 = UpSampling2D(2,  interpolation='bilinear')(conv_up_2_2)
conc_3 = Concatenate()([conv_2_2, up_3])

conv_up_3_1 = Conv2D(64, (3, 3), padding='same')(conc_3)
conv_up_3_1 = Activation('relu')(conv_up_3_1)

conv_up_3_2 = Conv2D(64, (3, 3), padding='same')(conv_up_3_1)
conv_up_3_2 = Activation('relu')(conv_up_3_2)



up_4 = UpSampling2D(2,  interpolation='bilinear')(conv_up_3_2)
conc_4 = Concatenate()([conv_1_2, up_4])
conv_up_4_1 = Conv2D(32, (3, 3), padding='same')(conc_4)
conv_up_4_1 = Activation('relu')(conv_up_4_1)

conv_up_4_2 = Conv2D(1, (3, 3), padding='same')(conv_up_4_1)
result = Activation('sigmoid')(conv_up_4_2)


model = Model(inputs=inp, outputs=result)
```

```
best_w = keras.callbacks.ModelCheckpoint('fcn_best.h5',
                        monitor='val_loss',
                        verbose=0,
                        save_best_only=True,
                        save_weights_only=True,
                        mode='auto',
                        period=1)

last_w = keras.callbacks.ModelCheckpoint('fcn_last.h5',
                        monitor='val_loss',
                        verbose=0,
                        save_best_only=False,
                        save_weights_only=True,
                        mode='auto',
                        period=1)


callbacks = [best_w, last_w]



adam = keras.optimizers.Adam(lr=0.0001, beta_1=0.9, beta_2=0.999, epsilon=1e-
08, decay=0.0)


model.compile(adam, 'binary_crossentropy')


batch_size = 2
model.fit_generator(keras_generator(train_df, batch_size),
            steps_per_epoch=100,
            epochs=200,
            verbose=1,
            callbacks=callbacks,
            validation_data=keras_generator(val_df, batch_size),
            validation_steps=50,
            class_weight=None,
            max_queue_size=10,
            workers=1,
            use_multiprocessing=False,
            shuffle=True,
            initial_epoch=0)
```

Основная часть системы:

```
colors = []
for k in range (100):
    R = rand.randint(0,255)
    G = rand.randint(0,255)
    B = rand.randint(0,255)
    colors += [[R,G,B]]

test = test_df[0:999]
fheight = 256
fwidth = 256
fourcc = cv2.VideoWriter_fourcc(*'MJPG')
out = cv2.VideoWriter('videos/out_modeln4.avi', fourcc, 20.0, (fwidth, fheight))
out1 = cv2.VideoWriter('videos/modeln4.avi', fourcc, 20.0, (fwidth, fheight))
miou = []
classes = []
ind = 0
obj = 0
mas = np.load(kol_path + '0.npy')
for t, t1 in test:
    im = cv2.imread(t)
    im_mask = cv2.imread(t1)
    im_write = np.array(im)
    # im = cv2.resize(im, (256,256))/255
    # im_write = cv2.resize(im_write, (256,256))

    num_og = mas[ind]
    ind += 1
    s1 = datetime.now()
    pred = model.predict(np.array([im]) / 255)
    t11.append(datetime.now() - s1)
    im_mask = im_mask[:, :, 0]
    tmp = np.array(im_mask)
    im_mask[tmp > 128] = 1
    im_mask[tmp <= 128] = 0

    mask1 = pred[0, ..., 0] > 0.9
    n1 = mask1 + im_mask
    n1[n1 > 0] = 1
    n2 = im_mask * mask1
    # print (n1, n2)
    miou.append((np.sum(n2) + 0.0000001) / (np.sum(n1) + 0.0000001))
```

```
s2 = datetime.now()
distance = ndi.distance_transform_edt(mask1, sampling=0.5)
exp_val = 50
expand = np.zeros((256 + exp_val * 2, 256 + exp_val * 2))
expand[exp_val:256 + exp_val, exp_val:256 + exp_val] = distance
local_maxi_coord = peak_local_max(expand, footprint=np.ones((20, 20)))
local_maxi = np.zeros((256, 256))

for coord in local_maxi_coord:
    local_maxi[coord[0] - exp_val, coord[1] - exp_val] = 1

markers = ndi.label(local_maxi)[0]
labels = watershed(-distance, markers, mask=mask1)

if fl:
    f = np.zeros(obj)
    for label in np.unique(labels):
        if label == 0:
            continue
        tmp = np.zeros((256, 256))
        tmp[tmp == 0] = -1
        tmp[labels == label] = last_frame[labels == label]
        elements, repeats = np.unique(tmp, return_counts=True)
        index = repeats.argmax()
        repeats[index] = 0
        index = repeats.argmax()
        elem = int(elements[index])
        if elem != 0:
            if f[last_color[elem]] != 1:
                class_color[label] = last_color[elem]
                f[last_color[elem]] = 1
            else:
                class_color[label] = obj
                obj += 1
        else:
            class_color[label] = obj
            obj += 1
else:
    obj = np.unique(labels).shape[0]

last_frame = np.array(labels)
last_color = np.array(class_color)
fl = True
# print(np.unique(labels))
```

```
    num_res = np.unique(labels).shape[0] - 1
    classes.append((min(num_og, num_res) + 0.0001) / (max(num_og, num_res) +
0.0001))

    for k in np.unique(labels):
        if k == 0:
            continue
        im_write[labels == k, :] = colors[class_color[k]]

    t22.append(datetime.now() - s2)
    cv2.imshow('frame', im_write)
    out.write(im_write)
    out1.write(im)
    # print(num_res, num_og, classes[len(classes)-1])
    k = cv2.waitKey(1)

    if k == 27:
        cv2.destroyAllWindows()
        break

out.release()
out1.release()
cv2.destroyAllWindows()
```