Санкт–Петербургский государственный университет

Павлова Екатерина Денисовна

Выпускная квалификационная работа

Решение задачи мониторинга состояния глубоководных животных и растений с использованием нейросетевых технологий

Уровень образования: бакалавриат
Направление 02.03.02 «Фундаментальная информатика и информационные технологии»
Основная образовательная программа СВ.5003.2017 «Программирование и информационные технологии»
Профиль «Автоматизация научных исследований»

Научный руководитель:
профессор, кафедра компьютерного моделирования
и многопроцессорных систем, д.т.н Дегтярев Александр Борисович

Рецензент:
доцент, кафедра вычислительных
методов механики деформируемого тела,
к.ф. - м.н. Седова Ольга Сергеевна

Санкт-Петербург
2021 г.

Содержание

Введение

На сегодняшний день актуальны морские исследования с целью мониторинга донных сообществ [37] и автоматического картографирования подводных ландшафтов [17].Задачи сегментации и распознавания под водой могут быть решены с помощью методов машинного обучения [24]. Подсчет подводных популяций необходим для проведения анализа экологического состояния среды

Для анализа данных по фото и видео- наблюдениям необходимо иметь размеченные наборы различных данных. Существующие базы данных, описывающие биологическое разнообразие глубоководного морского дна разрозненны и содержат мало структурированной информации. Также проблема создания обширных баз данных заключается в редкости многих донных обитателей, вследствие чего количество изображений определенного вида животных может быть очень мало. Именно поэтому решение задачи сегментации подводных сообществ целесообразно решать с помощью нейронных сетей решающих класс задач Few Shot Learning [19]. Особенность задач FSL заключается в том, что с их помощью возможно сегментировать объект имея всего несколько изображений каждого вида животных и растений. В ходе исследования нами были подготовлены небольшие наборы данных содержащих в себе 60 видов морских обитателей, а также проанализированы имеющиеся в свободном доступе датасеты.

Исследование донных сообществ также может заключатся в определении ареала обитания на карте и подсчетах объемов исследуемых объектов. 3D моделирование покрытия растениями и животными поверхностей по видео может помочь локализовать основные биотопы на картах [41]. Локализация подсчет покрытия могут быть использованы для дальнейших экологических исследований.

Была создана система, включающая в себя нейросетевые модели, необходимые для обработки видео, приложения для визуализации полученных данных и базу данных. Однако использование системы на компьютерах конечных пользователей, может быть затруднительно, так как требуется большое количество ресурсов. Также с ростом количества исследований, объемы ин-

формации, которую необходимо хранить и обрабатывать, могут существенно вырасти. Данные обстоятельства делают масштабируемость и распределенность необходимыми свойствами, которыми должна обладать система. Микросервисная архитектура подразумевает быструю портируемость и расширяемость, именно поэтому был рассмотрен именно этот вариант сервис-ориентированной архитектуры.

Большие системы перед вводом в эксплуатацию требуют тщательного тестирования. Если система является распределенной, то кроме корректности работы каждой из его компонент, необходимо учитывать их сетевые взаимодействия.

Существует ряд систем моделирования [16] (например, GNS3 [2], eveng [1]) , в которых можно развернуть сеть из виртуальных машин и тестировать приложение на ней. Но эти системы тестирования требуют от разработчика наличия знаний в области системного администрирования.

Приложения можно тестировать на реальных пользователях с помощью таких инструментов как Pumba [6] или Gremlin [3], но такое решение несет в себе риски потери интереса пользователей к приложению.

Рассмотрены способы моделирования масштабируемой сети. Такие технологии как MADT и L3NS предлагают рассматривать модель сети, состоящей исключительно из Docker образов-узлов и роутеров. Такой подход позволяет эмулировать распределенную сеть локально не вдаваясь в детали сетевых взаимодействий. Модель позволяет воспроизводить реально наблюдаемые разработчиками эффекты такие как дублирование, потеря, задержка или искажение пакетов данных.

В данной работе будет представлено решение задачи подсчета глубоководных малоподвижных обитателей с помощью нейросетевых технологий и его тестирование.

Постановка задачи

Целью работы является создание фреймворка решающего задачу разметки, сегментирования и определения на карте малоподвижных подводных обитателей. Для достижения поставленной цели необходимо решить ряд задач, которые в свою очередь можно разбить на 3 подмножества представленных на рис. 1.

Задачи первого этапа заключаются в формировании датасета. После нескольких итераций нахождения объектов, их обработки и выделения типов объектов, будет сформирована основная база данных. Она будет служить основой, содержащей по несколько фотографии целевых объектов, с помощью которых будет производиться сегментация на втором этапе работы.

На этапе предобработки и сегментации видеоряд проходит предобработку, которая заключается разбиение на кадры и улучшении качества изображения [27]. Нахождение и пред обучение few shot learning нейронных сетей на подводных объектах является одной из важных задач второго этапа. Эта задача имеет наибольшее влияние на качество результата работы всей системы.

После сегментации объектов начинается третий этап картографирования дна и локализации сегментированных объектов на построенной по видео 3D модели [35]. Построенная 3D модель позволяет оценить объемы объекта и его координаты относительно камеры и исследуемой точки дна [30].

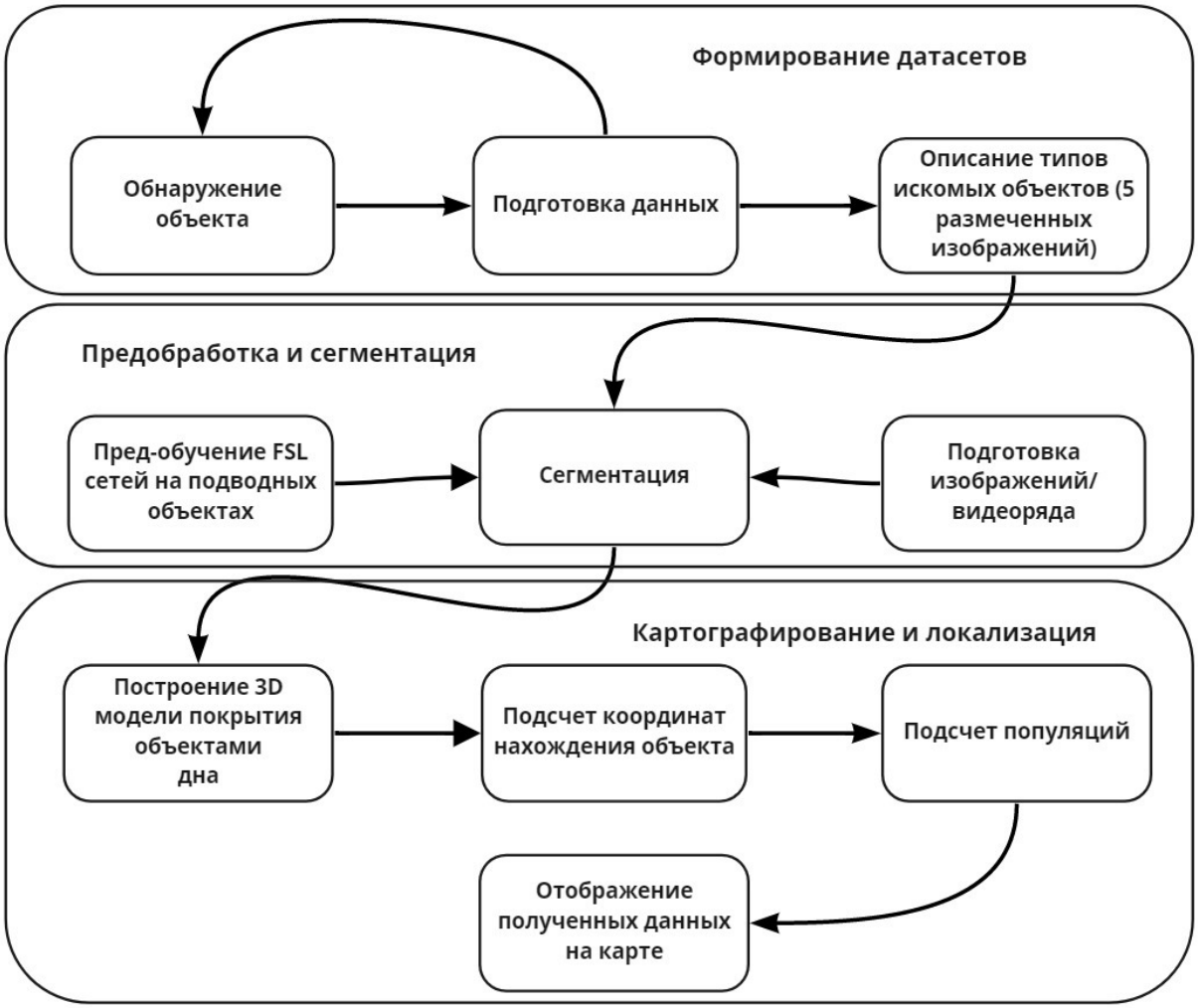


Рис. 1: Шаги для выполнения задачи изучения морского дна: формирование набора данных, предварительная обработка и сегментация, картографирование и локализация; и последовательность задач.

Обзор литературы

Создание системы, состоящей из нескольких элементов подразумевает затрагивание нескольких больших тем машинного обучения: сегментация для supervised и unsupervised задач, решение задач Few Shot Learning и таких тем как визуализация полученных данных на карте, использование датасетов глубины.

В статьях [22, 24, 41, 20, 37] описаны основные подходы, используемые для подводных исследований с помощью нейронных сетей. Более полное представление о работе с видеорядом под водой можно получить в [42, 17, 30]. Решение задачи улучшения качества подводного изображения с помощью сверточных нейронных сетей [27, 15] может впоследствии быть использовано как часть решения задачи сегментации, так как качество изображения является существенным фактором под водой.

Для решения класса задач Few Shot Learning в статье [28] описан датасет Fss-1000, состоящий из 1000 классов для few-shot сегментации. Архитектура сети, использованной для 5-shot сегментации описана в статье [14].

Для тестирование системы на распределенной сети были использованы [32, 25], в основе которых лежит [21, 16, 29, 38]. Методы виртуализации с помощью Doker описаны в [13, 31, 26].

Глава 1. Процесс обработки

1.1 Обзор

В статье [18] был описан подход для автоматического детектирования и сегментирования подводных крабов с целью их более точного кормления. Однако существенными недостатками данного решения являются небольшая глубина нахождения животных, их близкое расположение к камере. Во время глубоководной съемки рельеф местности может не позволять так близко приблизиться к объектам исследования.

Предлагаемое нами решение основано на создании каскада нейронных сетей, каждая из которых способна решать одну из поставленных задач. Основными задачами являются: улучшение качества изображения, сегментации и подсчета покрытия особями поверхности. Добиваясь наилучшего результата на каждом этапе [20], возможно достичь поставленной цели. Нейронные сети должны успешно справится с задачами unsupervised, semi-supervised или supervised сегментации на видео или фото, а также задачами super-resolution для предобработки данных.

1.1.1 Компоненты решения

Одним из важных критериев выбора компонентов алгоритма было ограничение на объем обучающего набора данных. Поэтому,в первую очередь, внимание было обращено на модели, способные выполнить задачу сегментации без крупномасштабных датасетов(например ImageNet и COCO). Модель должна выполнять ту же задачу, но по нескольким кадрам, то есть количество обучающих примеров может быть ограничено пятью и менее. Примером модели, обученной на few-short segmentation базе данных FSS-1000 и описанная в статье [28, 14]. Легкую расширяемость FSS-1000 также можно отнести к положительным характеристикам данного подхода.Архитектура модели представлена на Рис. 2

С задачей сегментации объектов эффективно [34] справляются сверточные нейронные сети. Для видео потока обычно такого рода задачи решаются путем раздельной обработки информации о внешнем виде и движении с

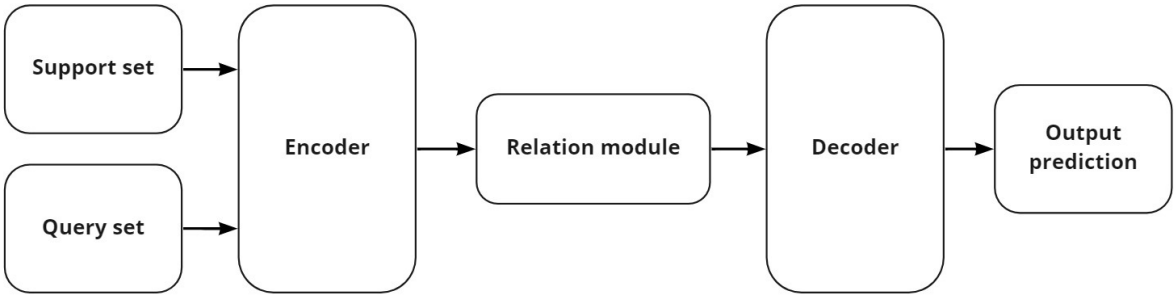


Рис. 2: Архитектура нейронной сети, обученной на базе данных FSS-1000.

использованием стандартных 2D сверточных сетей с последующим слиянием двух источников информации. Однако, новый подход - DC-Seg , превосходящий по скорости и точности накладываемой маски, описанный в статье [23], предлагает осуществлять сегментацию заметных объектов на видео с помощью 3D CNN. Такой подход ранее не был эффективно использован для решения проблем компьютерного зрения. Архитектура модели представлена на Рис. 3

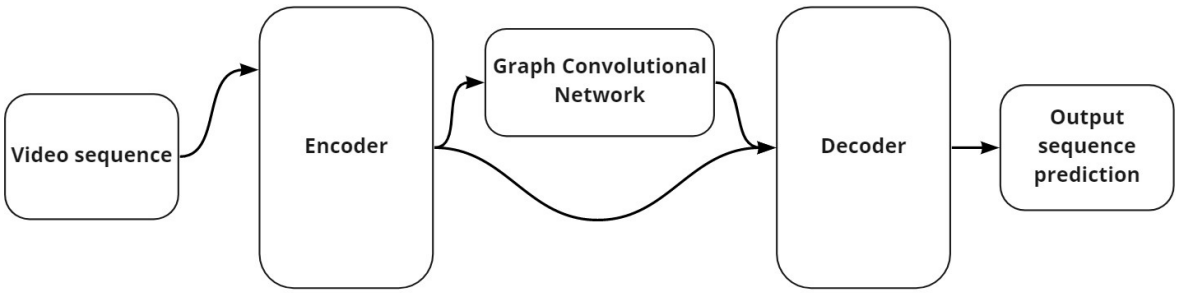


Рис. 3: Архитектура модели DC-Seg

Из за сложного рельефа или слишком близкого расположения друг к другу объектов их может быть сложно сегментировать, а вследствие этого и учесть при подсчете. Поэтому проблема сегментации фонового или частично скрытого объекта также актуальна для видео мониторинга подводного

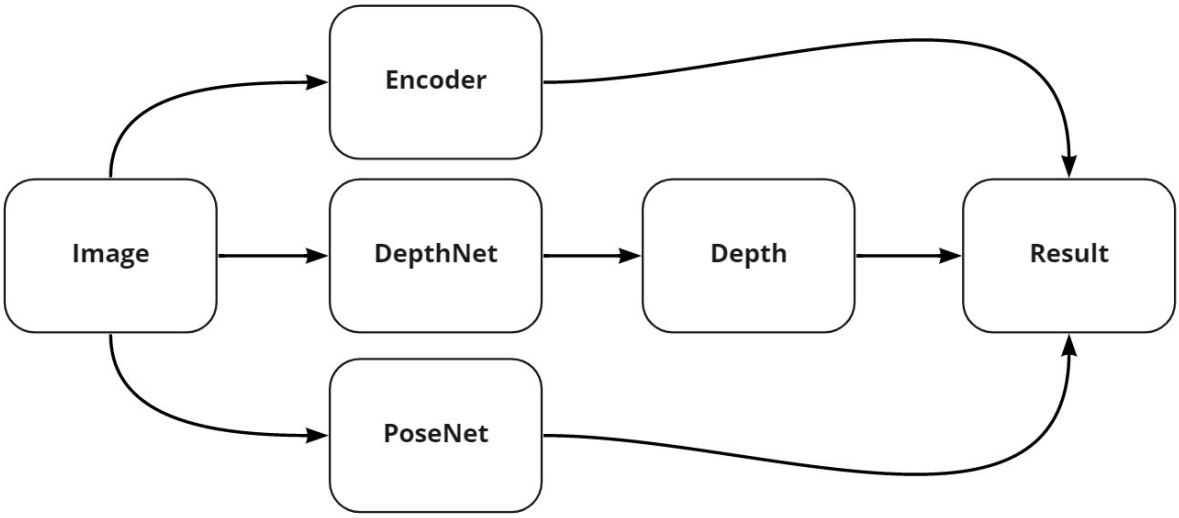


Рис. 4: Архитектура фреймворка решающего задачу single-view reconstruction

биотопа. Semi-supervised модель [42], нацелена на сегментирование определенного объекта по всей видеопоследовательности на основе маски объекта, заданной на первом кадре. Такой метод может быть полезен для сегментации интерактивных объектов.

Ввиду особенностей видео под водой, мутное изображение- одна из самых часто встречающихся преград для точного наложения маски на объект. Поэтому также был опробован метод, когда видеоряд разбивается на кадры, к каждому из которых применяется SISR [23], генеративная super-resolution модель для увеличения четкости кадра. Для подсчета объема покрытия объектами дна необходимо составить карту глубины для ландшафта и сегментированных объектов. Так как мы можем использовать только одну камеру для подводной съемки, нам нужно решать задачу Single-view reconstruction. Фреймворк, описанный в статье [39] решает данную задачу с помощью комбинации 3 нейронных сетей : DepthNet, PoseNet and FeatureNet для предсказания карты глубины, движения и изучения особенностей соответственно. Архитектура фреймворка представлена на Рис. 4

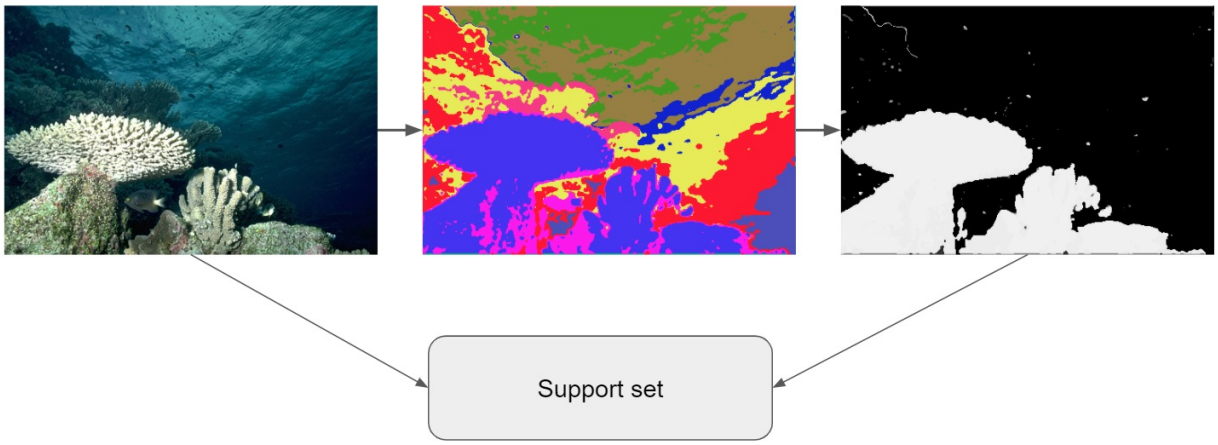


Рис. 5: Подготовка данных для support set: изменение разрешения и размера, семантическая сегментация, выделение необходимых масок.

1.1.2 Подготовка данных

Перед тем как решать задачу сегментации, необходимо создать набор классов разных видов животных и растений. На рис. 1 после обнаружения объекта наступает этап подготовки данных без которого невозможно перейти к этапу сегментации. Это объясняется тем, что нейронной сети нужны небольшие наборы данных с целевыми объектами, которые должны быть сегментированы на изображении.

Одна из самых известных открытых баз данных - Fish Recognition Ground-Truth data [7], использованная в проекте Fish4Knowledge для решения задач детектирования и распознавания [8]. Однако она содержит только 15 видов рыб, обитающих около кораллового рифа. Другие же датасеты [9, 10] содержат около 500 видов, но в качестве выделения объектов используются прямоугольники, в связи с чем часто плавники и части тела могут быть обрезаны. SUIM Dataset for semantic segmentation of underwater imagery [22] мог бы облегчить создание датасета, но содержит 8 существенно больших, чем требуется классов, таких как human divers or aquatic plants and sea-grass.

Для тестирования решения задачи FSL сегментации различных морских обитателей была создана выборка из 6 изображений для каждого вида [**?**]. Для данного датасета были произведено изменение размера исходного изображения до размера 224 X 224 пикселя и создание черно-белой маски изображения такого же размера.

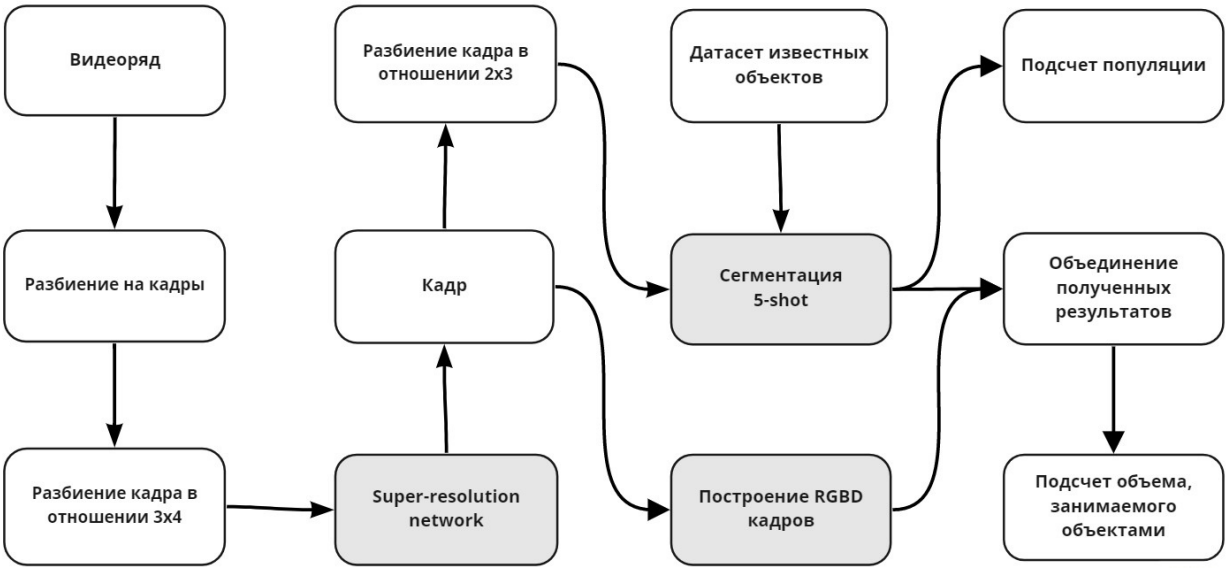


Рис. 6: Этапы обработки видеопоследовательности.

Первый этап на рис. 1 заключается в автоматизированном создании масок изображений и подготовки базы данных. С помощью нейронной сети решающей задачу semantic segmentation на каждом из кадров сегментируются разного рода объекты. После сегментации к разным классам объектов относятся полученные маски. Процесс создания датасета для дальнейшего его использования в качестве support set для обобщения на неувиденные образцы из query set представлен на рис. 5.

1.2 Реализация

Таким образом, на первом этапе формирования датасета с помощью модели DC-Seg на видео материалах сегментируются разного рода объекты, которые впоследствии будут размечены и сформированы в классы по 5 и больше изображений(рис. 7). На втором этапе происходит предобработка видео – разбиение на кадры и применение super-resolution модели SISR. С помощью заранее подготовленного на первом этапе датасета разбитый на кадры видеоряд сегментируется с использованием модели 5-shot, предобученной на датасете FSS-1000. Третий этап начинается с построения карты глубины для входящего видеоряда с помощью фреймворка описанного выше. Полученные RGBD кадры объединяются с сегментированными ранее объектами, что дает возможность не только посчитать количество объектов, но и вычислить ими

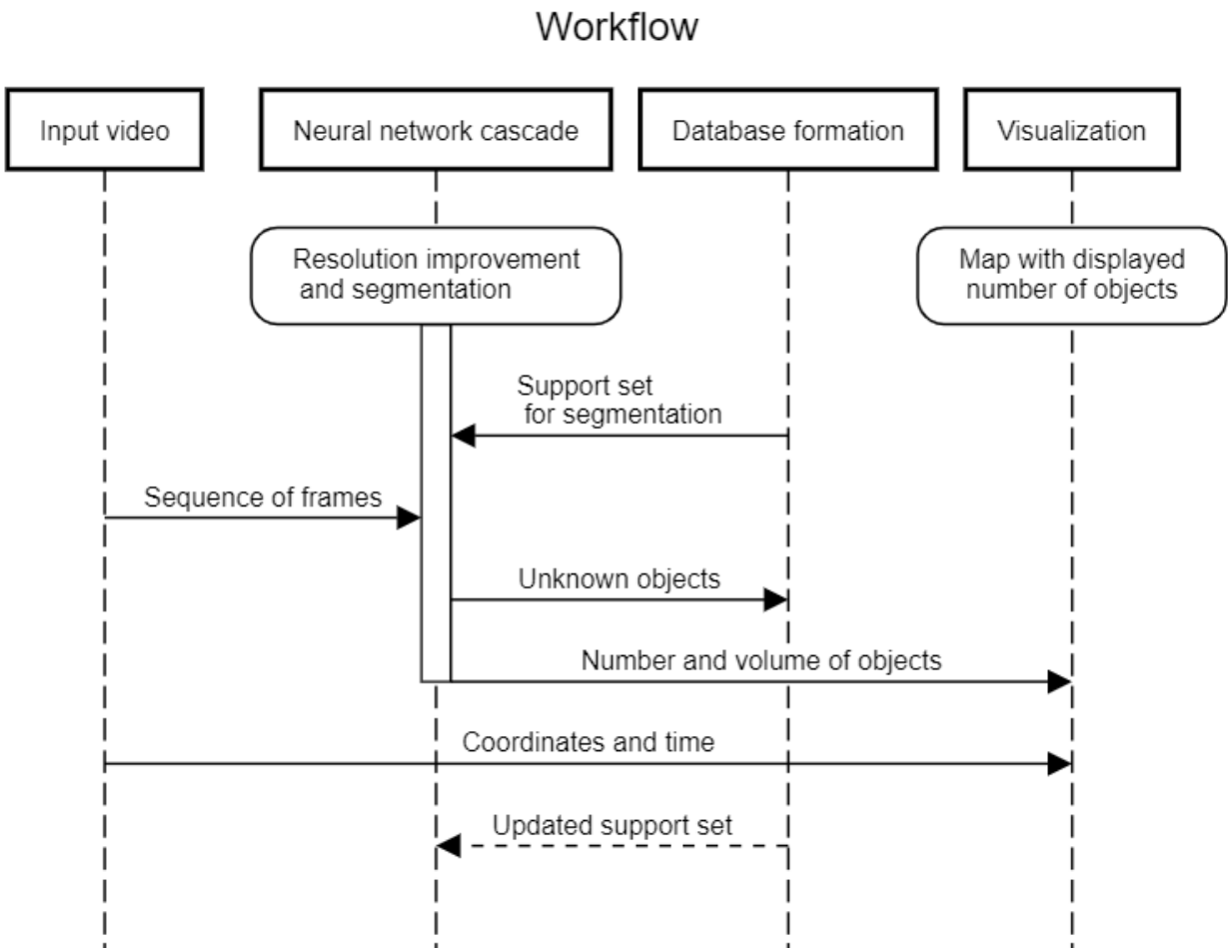


Рис. 7: Схема обработки входящей видеопоследовательности. Видео разбивается на кадры, каждый кадр обрабатывается нейронными сетями. Ранее неизвестные объекты хранятся в наборе данных. Координаты, время и результаты подсчета синхронизируются и визуализируются на карте.

занимаемый объем. Полученное решение представлено на рис. 6.

Видеоряд подводной съемки разбивается на кадры, которые в свою очередь проходят предобработку и с помощью нейронной сети увеличивают разрешение изображения. Сегментация объектов из базы данных производится на кадрах прошедших предобработку. Более подробное описание использованных нейронных сетей представлено в таблице 4.

Unsupervised task видео сегментации заключается в сегментации всех пикселей принадлежащих “выдающийся” объекту, при этом категории объектов не известны априори. Решение такой задачи необходимо для автоматического сбора подводных баз данных. 5-shot модель использует в качестве изображений целевых объектов для сегментации элементы созданной ранее базы данных. С помощью SISR каждый кадр изображения увеличивается в 4

Модель	Параметры	Pre-train модель	Датасет	Применение сегментации
3DC-SEG	Total params: 103,244,936 Trainable params: 103,244,936	+	COCO, YouTubeVOS, DAVIS'17	Видео сегментация (unsupervised task)
5-shot	Total params: 32,833,025 Trainable params: 32,833,025	+	FSS-1000	FSL сегментация
SISR	Total params: 1,944,579 Trainable params: 1,940,227	-	USR-248	Single Image Super-resolution

Таблица 1: Рассмотренные нейросетевые модели.

раза.

1.3 Визуализация

Графическое отображения результатов создается с помощью open-source приложения для визуализации крупномасштабные наборы данных геолокации [15] – Kepler.gl [11]. Для построения рельефа на карте мира в первую очередь необходимо получить высотные модели. Были использованы глобальные датасеты [12] SRTM 30m Global 1 arc second V003 разработанная NASA и NGA размещенная в Amazon S3 и SRTM 90m Цифровая база данных высот v4.1 разработанная CGIAR-CSI. С помощью открытых библиотек основанных на GDAL растровый файл преобразуется в подходящий для Kepler.gl формат. Данные полученные из видео материалов подводного исследования, такие как глубина, широта, долгота, площадь занимаемой объектом поверхности и временная метка, строятся поверх глобальные цифровые модели местности. Также при визуализации данные трансформируются из из собственной системы координат набора данных в CRS84. Процесс создания отображен на рис. 8.

1.4 Полученные результаты

Одной из главных проблем с которыми мы столкнулись являлось отсутствие подводных датасетов для сегментации большого количества разных объектов. Решение данной проблемы стало частью архитектуры нашей системы, с помощью которого можно по исходной видеопоследовательности сформировать необходимый датасет.

Рис. 8: Процесс визуализации карты глубины и полученные результаты.

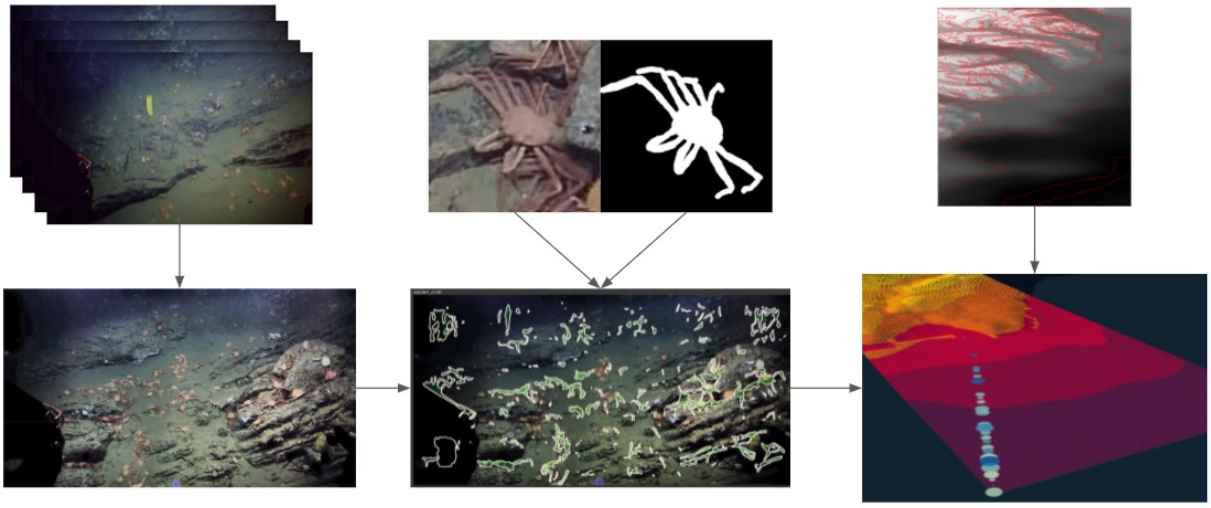


Рис. 9: Блок-схема, показывающая результаты каждого этапа.

Отсутствие большого количества вычислительных ресурсов и ограниченность данных способствовало выбору в пользу пред обученных моделей или моделей, не требующих долгого обучения. Однако веса super-resolution модели были получены самостоятельно и опубликованы в открытом доступе.

На сегодняшний день существует большое количество решений задачи семантического сегментирования, но подводные видео обладают рядом особенностей, таких как : мутность воды, недостаток света, частичная видимость и тд. Это накладывает дополнительные ограничения и создает сложности в выборе архитектуры модели.

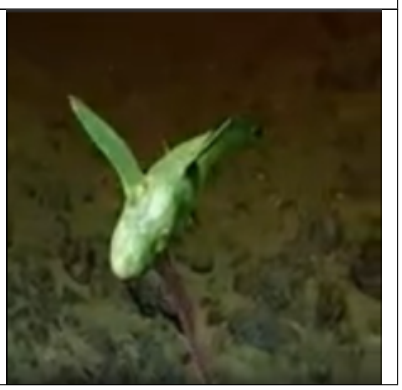
Полученное решение представлено на рис. 9. На каждом из этапов ре-

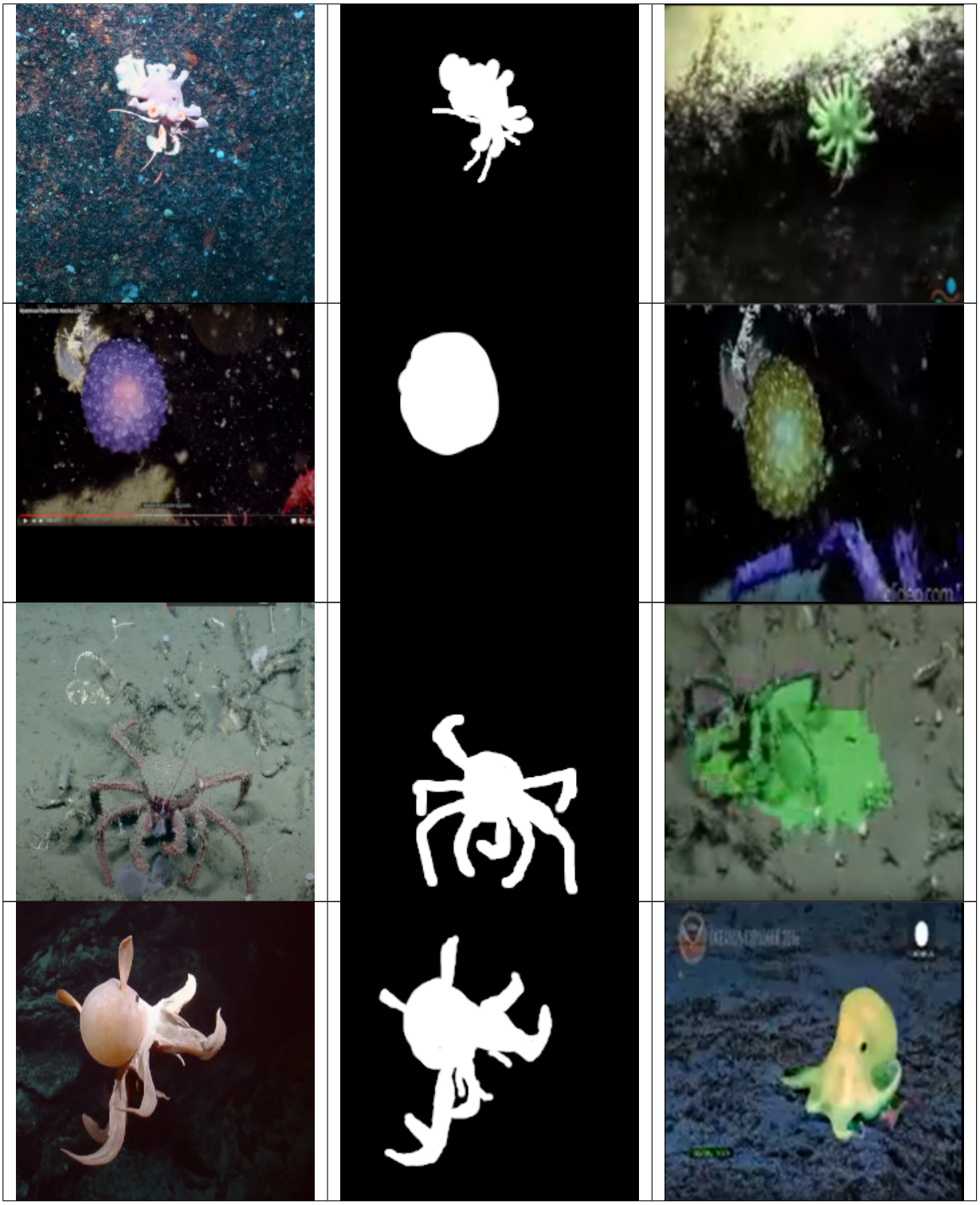
Метод	Видео	SISR	5-shot	Общее время	Результат
Представленное решение	144s (29,97 fps)	1230s (14 fps)	2723s	3953s	csv + карта
	1 кадр	40-50s	9-10s	50-60s	
Стандартный метод	1 кадр			∼60s	Количество объектов

Таблица 2: Время обработки видеопоследовательности.

шения используются наиболее современные методы и модели. использованные нейронные сети и инструменты способны решать поставленные задачи быстро. В таблице 2 представлено примерное время работы полученной автоматической системы в сравнении с классическими методами исследованиями дна.

В таблице 3 представлены некоторые результаты и входные данные для модели few shot learning. Хочется отметить, что качество видеосъемки и прозрачность окружающей среды существенно влияют на итог работы нейронной сети. Так, например, краб покрытый слоем песка не был сегментирован на большинстве кадров. Аналогичные результаты были получены с видеозаписью, где осьминог плавает у поверхности воды. Блики солнца создают неравномерные тени на изображение, что усложняет задачу.

Изображение	Черно-белая маска	Результат
		



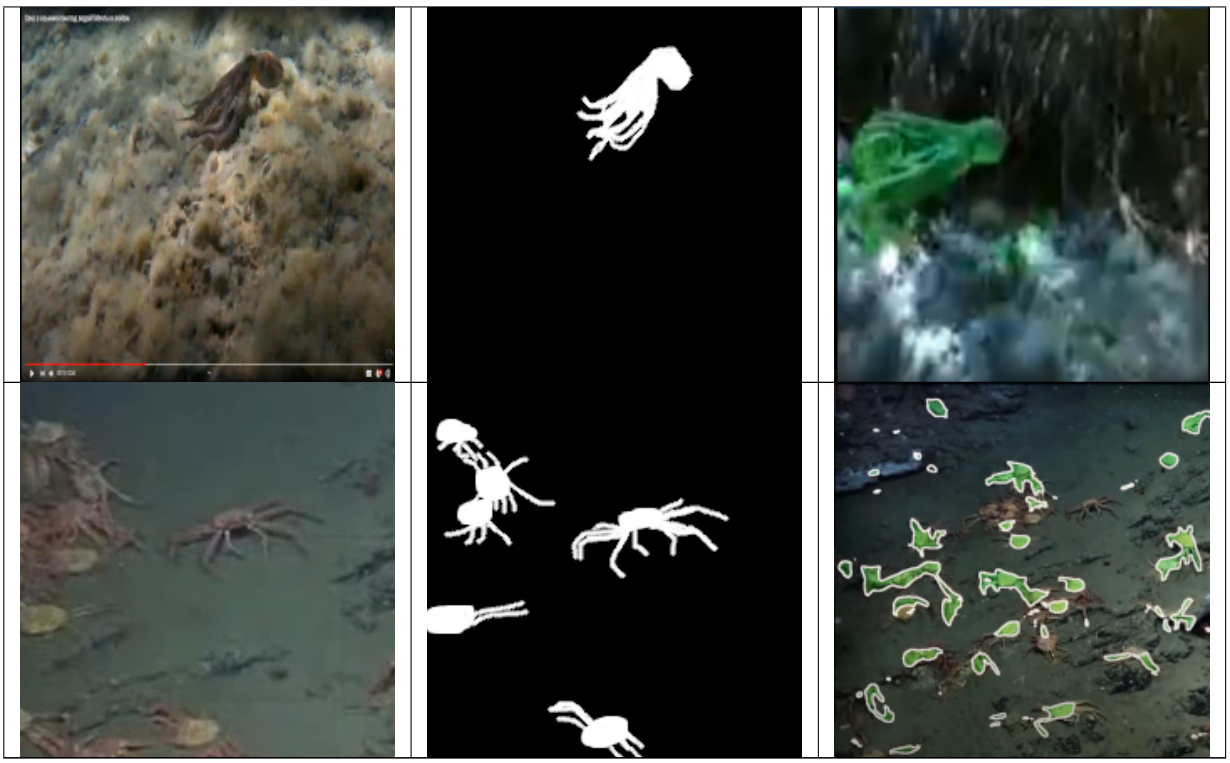


Таблица 3: Демонстрация вхожных данных и результатов работы нейросетевой модели few shot learning. 1.Элемент support set, 2. Маска с выделенным объектом, 3. Кадр из результирующей видеозаписи.

Был создан датасет из 60 животных, некоторые из которых представлены в таблице. Каждый из объектов датасета может являться элементом support set для FSL модели. Животные способные маскироваться (некоторые виды крабов), быстро менять форму или быстро передвигаться (уплывающий осьминог) в некоторых случаях сегментируются неточно.

1.5 Сформированная модель

С помощью представленной архитектуры можно успешно решать задачу разметки и детектирования малоподвижных подводных обитателей. Новые алгоритмы улучшения изображений и сегментации подводных объектов могут быть использованы для автоматизации мониторинга и подсчета популяций. По сравнению с традиционными методами изучения морской флоры и фауны автоматизированный подход может значительно улучшить качество данных и сократить количество затрачиваемых ресурсов. Полученное решение может быть использовано как основа для создания обширных баз данных для последующего обучения статичных сегментационных моделей.

Глава 2. Система

Архитектура, описанная в первой главе, может быть развернута на одной машине клиента. Однако, обычно большинство исследований проводятся несколькими исследователями или даже несколькими компаниями. Также при значительном увеличении количества данных для отображения может возникнуть нехватка мощностей клиентской машины. Это может вызвать временные задержки и ошибки отображения данных. Представленная архитектура подразумевает периодическое дообучение моделей и хранение промежуточных данных в виде результатов семантического сегментирования. Поэтому требуется наличие дополнительных вычислительных ресурсов и памяти для нормального функционирования архитектуры.

Также возникает проблема синхронизации данных, полученных в ходе разных подводных исследований. Данные, хранящиеся на разных устройствах, могут быть намного полезнее, если их агрегировать в одном месте. Доступ к хранилищу можно получить из любой точки мира и тем самым внести свой вклад в морские исследования.

Создание распределенного приложения продиктовано необходимостью использования большого количества ресурсов и обладания полной информации о подводных исследования по всему миру. Для обучения и обработки данных нейросетевые модели нужны не только данные, но и вычислительные мощности, способные в короткие сроки дообучать модели. Одновременно с этим существует необходимость взаимодействовать с конечными пользователями. Количество запросов к системе может существенно вырасти с ростом пользователей. Следовательно необходимо масштабируемое решение.

2.1 Архитектура

Архитектура системы была выбрана с учетом возможности легкой замены ее компонентов и масштабируемости системы в будущем. Микросервисная архитектура подразумевает разделение функций приложения на сервисы , каждый из которых разворачивается в контейнере. Технология контейнеризации позволяет эффективно использовать ресурсы и обеспечивает портативность системы. Для взаимодействия между контейнерами используется API.

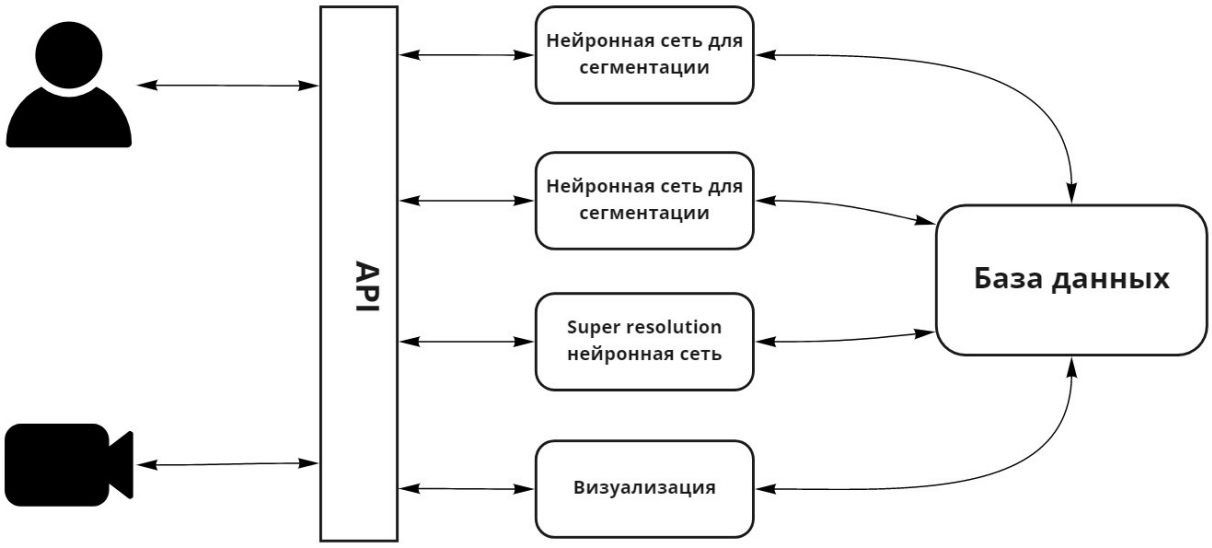


Рис. 10: Микросервисная архитектура системы.

На рисунке представлен микросервисная архитектура состоящая из:

- API;
- 4 сервисов;
- базы данных.

2.2 Алгоритм взаимодействия

На Рис. 11 изображен алгоритм взаимодействия пользователя и приложения. Конечным результатом работы приложения является визуализация координат, полученных с камеры, и результатов обработки видеозаписей. Пользователь делает запрос к API для предоставления результатов работы системы. Микро Сервисную архитектуру можно условно разделить по их функциональности на : сервисы для визуализации, сервисы для работы с нейросетевыми моделями и сервисы для взаимодействия с базой данных. Предыдущие видеопоследовательности и треки координат хранятся в базе данных и предоставляются по запросу пользователя.

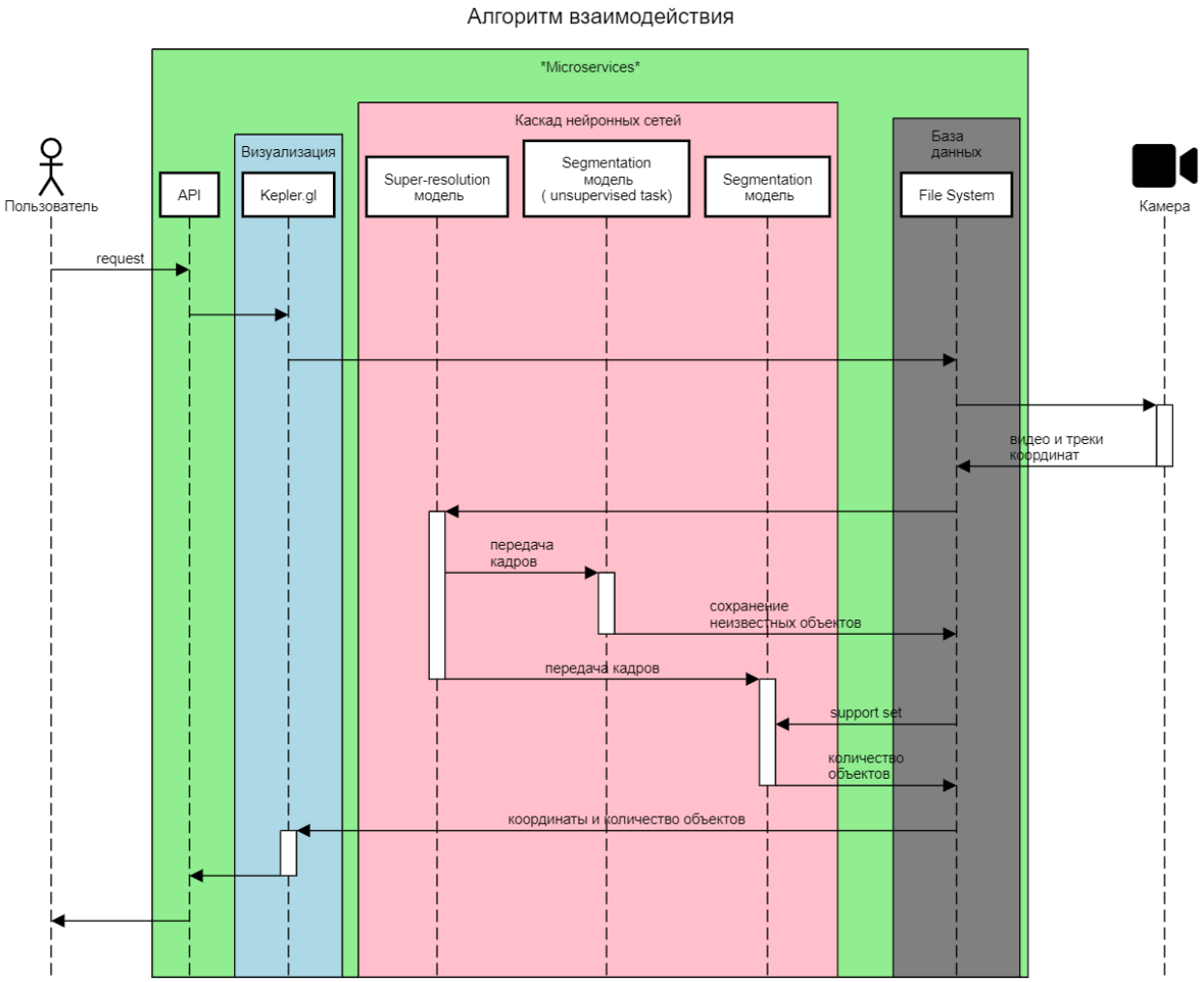


Рис. 11: Схема работы компонентов системы.

2.3 Обработка ошибок

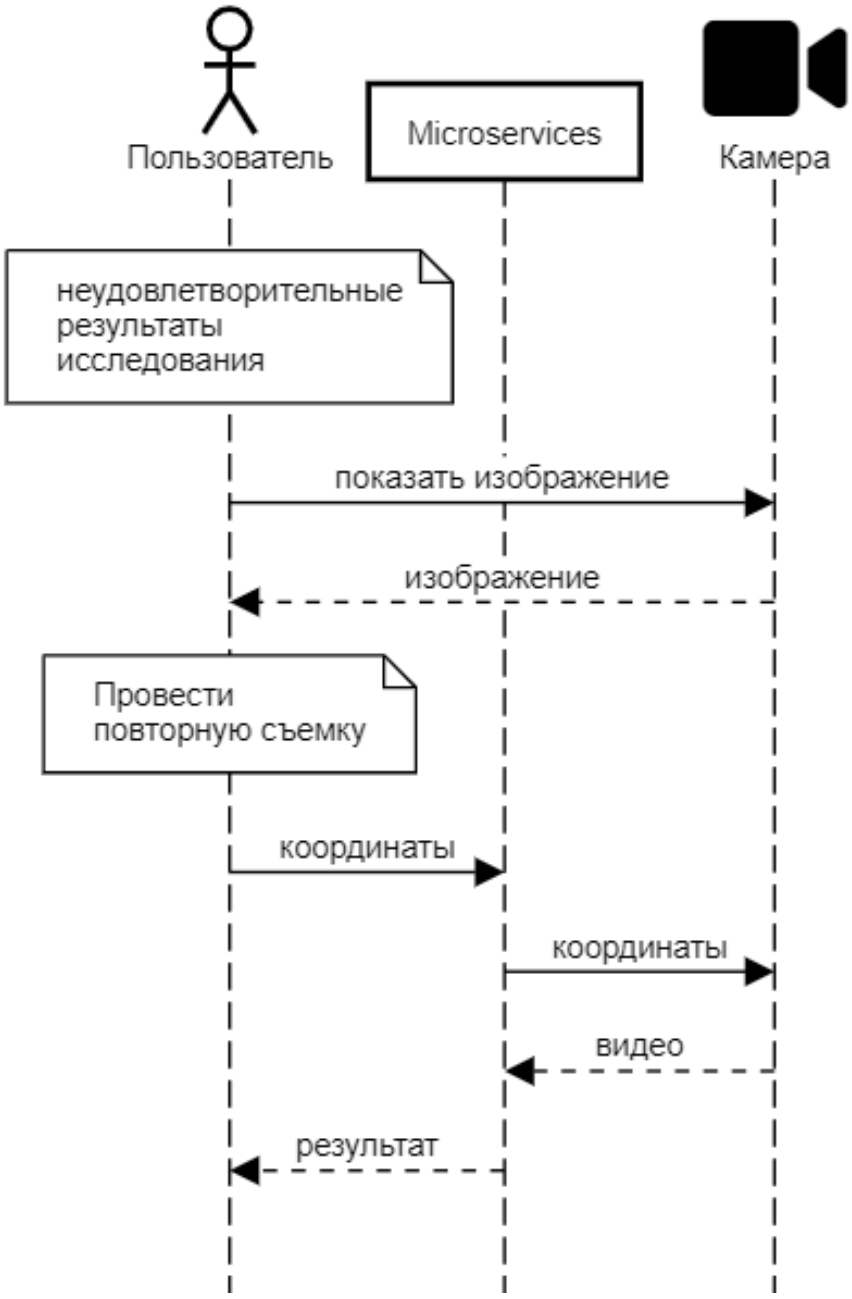


Рис. 12: Схема работы компонентов системы при ошибке видеосъемки.

Ошибки видеосъемки естественное явление в подводных условиях. Искажение и мутность изображения могут существенно ухудшить результаты. Поэтому архитектура должна быть гибкой и учитывать возможность недостаточно качественных результатов и предоставлять пользователям возможность запросить повторную съемку конкретных мест или треков целиком. Пример взаимодействия пользователя с системой при ошибках съемки представлен

на Рис. 12.

2.4 Полученные результаты

Представленная микросервисная архитектура системы способна использовать распределенные вычислительные ресурсы, что позволяет сократить время обучение моделей и обработки видеопоследовательностей. Также разделение функций приложения между сервисами способствует легкой масштабируемости решения. Технология контейнеризации упрощает процесс портирования системы и взаимодействия с ней. В архитектуре учтены возможности некачественной видео съемки, что позволяет пользователям уточнять результаты, посредством повторных запросов к системе.

Глава 3. Тестирование

На Рис. 10 представлена архитектура системы, каждый элемент которой может находиться на разных узлах компьютерной сети в силу необходимости использования большого количества ресурсов. В связи с этим возникает проблема синхронизации работы всего приложения. Необходимо учитывать риски возможных нарушений передачи данных по распределенной сети. Для того чтобы понаблюдать за работой системы в экстремальных условиях необходимо либо полностью реализовать сети, развернув части приложения на разных физических узлах, либо воспользоваться существующими решениями для проведения тестирования. По результатам которого можно устранить найденные уязвимости и проверить устойчивость приложения относительно нестабильной работы сети.

Существует около двух десятков эмуляторов распределенных сетей [40]. Большинство из них бесплатны и размещены в открытом доступе (в том числе на GitHub). Кроме того, HP, Huawei, Cisco имеют собственные сетевые эмуляторы, ориентированные на работу с оборудованием конкретного производителя. Рассмотрим наиболее известные среды моделирования сетей. Основными характеристиками для сравнения были выбраны(Табл. **??**): простота использования, масштабируемость и универсальность- таблица. Простота использования инструмента оценивается по следующим критериям: формат конфигурации, наличие наглядной визуализации, автоматическое развертывание и запуск распределенных приложений, возможность доступа приложений в сети интернет.

Не существует универсального решения для эмуляции систем большого масштаба. В некоторых случаях предпочтение отдается моделированию деталей процессов сетевого уровня L2, в других же случаях взаимодействие рассматривается на уровне модельных абстракций. Технологии L3NS [4] и MADT [5] позволяют сохранив всю сложность рассматриваемых сервисов максимально упростить создаваемую сетевую абстракцию, предоставив возможность моделирования как на одном так и нескольких вычислительных узлах и платформах.

Таблица 4: Сравнительная таблица существующих технологий моделирования

Опции	L3NS	Madt	Netkit/ Kathara	Mininet	Marionnet	zimulator (VNUML)	Eve-NG	NS3	GNS3	Omnet++ / Omnest
Формат конфигурации сети	Python Script	Python Script/ JSON	BASH script / special config	Python API	project file with conf of vm	VNUML Script	labs	c++ script	virual network editor	NED topology description language
Использование облегченной виртуализации [13]	+	+	+	- VirtualBox, Qemu, VMware	- VirtualBox, VMware	-	-	+	-	-
Визуализация	-	+	+ Netkit Lab Generator	+	+	-	+	+	+	+
Автоматически развертывать и запускать распределенные приложения	+	+	Приложение должно быть правильно установлено в хост-системе	+	+	+	+	+	-	-
Возможность масштабирования системы за счет использования более 1 физического узла для развертывания	+	-	-	+	-	-	+ for a fee	+	+	+
Уровень сетевой симуляции	L3	L3	L3	L2	L2	DES*	L2	L2	L2	DES
Подключение внешних сетевых узлов	+	-	-	+	+	-	+	-	+	-
Возможность доступа к приложениям в Интернете	+	+	+	+	+	-	+	+	+	+
Платформы, на которых могут работать смоделированные приложения	Linux	Linux	all platforms	Unix/ Linux	GNU/ Linux	Linux	VM based Windows, Linux, network devices	Linux, MacOS, FreeBSD	VM based Windows, Linux, host device	all platforms where a modern C++ compiler is available
Open source	MIT	Academic Public License	GPL3	MIT derevative	GPL	GPL3	Academic Public License	GPL2	GPL	Academic Public License

DES – Discrete-event simulation, not bound to networking level.

3.1 Компоненты решения

Для моделирования виртуальной компьютерной сети с помощью MADT или L3NS используется платформа контеринезации Docker [32], а при запуске модели на кластере для моделирования IP подсетей используется WireGuard [29]. Кроме того применяется набор утилит FRRouting, которые позволяют реализовать протоколы динамической маршрутизации, и оболочка Tcconfig, позволяющая динамически управлять качеством работы сети и использующаяся для улучшения реалистичности работы модели. Алгоритм разделения гиперграфа KaHyPar применяется для организации распределенного моделирования виртуальных компьютерных сетей [36]. Библиотеки jgraph, Flask и система передачи сообщений zmq используется для работы пользовательского интерфейса.

3.2 Отличия возможностей моделирования с помощью библиотеки L3NS и комплекса программ MADT

В то время как MADT позволяет быстрее ознакомиться с концепцией предложенной архитектуры и ставить эксперимент визуально, L3NS обладает функционалом для более быстрого и гибкого описания модели взаимодействия узлов и внедрения в CI экосистему.

Код L3NS не зависит от MADT. Основными функциональными особенностями являются логика распределения ip-адресов. Адреса распределяются по мере описания “лаборатории”, благодаря чему ip-адреса одних узлов можно использовать при описании других (например, сразу передать клиентам адрес сервера). Также в L3NS сильно упрощена работа с докер-сетями: их ip-адреса, как и ip-адреса узлов в них задаются напрямую, в то время как в MADT каждому интерфейсу в сети выдается второй ip-адрес. При описании элементов сети, в L3NS могут быть использованы разные стратегии запуска и остановки узлов. Например, используя подмодуль ldc- local docker container, ноды будут запускаться как Docker контейнеры, а используя подмодуль cluster, контейнеры будут запускаться на удаленных машинах. L3NS предоставляет возможности настройки удаленных машин по ssh. Другими словами существует возможность получить доступ к клиенту на определенном хосте, и

,например, просмотреть список образов, используемых на нем. Для запуска модели сервисной системы на нескольких виртуальных машинах нужно:

- Настроить ssh ключи;
- Установить Docker;
- Установить зависимости в python;
- Установить WireGuard.

В L3NS Реализована поддержка параллельного запуска контейнеров, с помощью ThreadPoolExecutor, что на порядок ускоряет запуск “лабораторий”, а также нетворкинг через докер сварм по сравнению с MADT. Подход, основанный на создании единого решения в форме библиотеки для программиста-пользователя, показал большую гибкость в задачах масштабирования системы, чем комплекс программ MADT, разделяющий задачи описания, запуска и конфигурации “лабораторий”.

3.3 Архитектура

Классическим способом моделирования вычислительных узлов сети является применение гипервизоров, управляющих виртуальными машинами. Ядра операционных систем разделены, это предоставляет изоляцию приложениям. В случае моделирования сети взаимодействующих приложений-сервисов, узлы и приложения в них не нуждаются в изоляции на уровне ядра ОС, изолированности сетевого стека, который будет использовать вычислительный узел, достаточно [21].

Ядро Linux предоставляет инструмент (Namespaces), который позволяет изолировать низкоуровневые окружения. Контейнером называют изолированный набор пространств имен, поверх которого запущено приложение. Платформа контеринизации Docker позволяет создавать такие окружения по заранее определенным шаблонам и запускать приложения в них. Кроме того с помощью Docker можно объединять сетевые интерфейсы, которые находятся в разных сетевых пространствах имен, виртуальным мостом в одну общую IP подсеть, хостами в которой являются контейнеры. MADT и L3NS позволяют

заранее задать топологию сети и параметры запуска контейнеров. Сеть запускается с помощью Docker и получается виртуальная модель сети, которая полностью работает на одном компьютере.

В случае если моделируемое приложение требует большого количества ресурсов, целесообразно разделить моделирование на несколько компьютеров. При этом решается задача эффективного разделения нагрузки по хостам для минимизации задержки в виртуальных сетях, соединяющих контейнеры, находящиеся на разных хостах. В каждом из контейнеров, запущенных в рамках модели, смонтирован unix-сокет. Это позволяет передавать сообщения в систему мониторинга, даже если сетевые интерфейсы контейнера заблокированы или имеют большую задержку, таким образом обеспечивая стабильный мониторинг. С помощью веб интерфейса можно в режиме реального времени наблюдать за состоянием модели и для каждого из узлов интерактивно управлять параметрами сети, контролируя [33] задержки, потери, дублирования пакетов.

3.4 Алгоритм моделирования

Работа происходит по следующему алгоритму:

1. С помощью API на языке Python создается
2. “Лаборатория” - описание модели сети, включающее в себя список узлов сети, их распределение по подсетям и параметры запуска. “Лаборатория” запускается, создавая сетевые узлы на базе Docker контейнеров и сетевые связи между ними
3. Отображение эмулируемой сети доступно разработчику в веб-интерфейсе для проведения экспериментов и наблюдения за их ходом.

3.5 Моделирование приложения с клиент-серверной архитектурой

Рассмотрим как с помощью предложенной концепции выполняется построение сети с набором роутеров и динамической маршрутизацией. Сеть можно представить в виде дерева, узлы которого – роутеры и клиенты.

Построение такой сети выполняется по следующему принципу: граф расширяется с помощью нескольких итераций добавления подсетей для дочерних узлов, в конце каждой из которых массив внешних узлов обновляется. Для всех узлов, кроме внешних, настраивается работа протокола динамической маршрутизации. Это позволит трафику проходить через дерево.

Каждому дочернему узлу графа ставится в соответствие Docker-образ, который будет использован при создании узла клиента.

Еще один шаг, который необходимо сделать для корректной работы нашей виртуальной сети – указать адрес сервера для каждого клиента, чтобы они могли взаимодействовать.

При запуске модели появляется возможность задавать неполадки сети вручную и наблюдать за тем, как они влияют на работу приложения.

В MADT gосле сохранения конфигурации построенной сети можно переходить в графический интерфейс и наблюдать за работой приложения.

Чтобы настроить состояние сети, необходимо перейти к любому узлу и установить параметры помех с помощью открывшегося окна, а затем нажать кнопку tcset. Например, чтобы роутер терял 50 процентов пакетов, которые ему приходят, нужно установить соответствующее значение поля loss.

Задав неполадки для нескольких узлов, можно наблюдать за результатом на графе. Цвет узла зависит от того, какой статус у пришедшего сообщения. Зеленый цвет сигнализирует о нормальной работе, жёлтый - о неполадках, красный - о критических проблемах, а фиолетовый - о неожиданных исключениях.

3.6 Полученные результаты

На первом этапе моделирования был создан образ каждого из компонентов системы. Следующим этапом являлось создание лаборатории посредством описания сети на Python. После успешного запуска лаборатории можно взаимодействовать с сетью с помощью пользовательского интерфейса.

Таким образом MADT и L3Ns могут быть использованы для:

- проверки устойчивости приложения относительно нестабильной работы сети;

- изучения влияния структуры сети на работу каждой части приложения;
- проверки наличия уязвимостей сетевого плана во взаимодействии частей системы;

Были рассмотрены случаи увеличения времени задержки и потери пакетов. Визуализировано состояние работающего в модели IP-сетей распределенного приложения в реальном времени.

Выводы

Описанная в первой главе архитектура была реализована и протестирована на более чем 20 видеозаписях. Был создан датасет состоящий из 60 видов животных, который может быть использован в дальнейших исследованиях. Время обработки 1 кадра нейросетевыми моделями сравнимо с временем, затрачиваемым исследователями на ручной подсчет. Однако, если рассматривать всю видеопоследовательность целиком, представленный метод может выдавать результаты более оперативно и качественно.

Система из второй главы учитывает особенности работы с нейросетевыми моделями, а именно необходимость использования большого количества вычислительных ресурсов. Разработан и представлен на рисунке алгоритм взаимодействия компонентов системы.

С помощью технологий из третьей главы было проведено исследование поведения распределенной сети, полученной вследствие развертывания элементов системы на разных узлах. Были рассмотрены ситуации потери пакетов, увеличение времени задержки передачи, дублирования пакетов. Впоследствии были учтены возможные ошибки работы системы из за нестабильной работы сети.

Таким образом, созданная система выполняет все поставленные задачи, а именно решает задачу разметки, сегментирования и определения на карте малоподвижных подводных обитателей. Распределенная архитектура приложения способствует более быстрому получению результата.

Заключение

С помощью представленной архитектуры можно успешно решать задачу разметки и сегментации малоподвижных подводных обитателей. Новые алгоритмы улучшения изображений и сегментации подводных объектов могут быть использованы для автоматизации мониторинга и подсчета популяций. По сравнению с традиционными методами изучения морской флоры и фауны автоматизированный подход может значительно улучшить качество данных и сократить количество затрачиваемых ресурсов. Полученное решение может быть использовано как основа для создания обширных баз данных для последующего обучения статичных сегментационных моделей.

Список литературы

[1] EVE-NG webpage (Last accessed 20 June 2020), https://www.eve-ng.net/

[2] GNS3 webpage (Last accessed 20 June 2020), https://www.gns3.com/

[3] Gremlin webpage (Last accessed 20 June 2020), https://www.gremlin.com/

[4] L3NS GitHub repository (Last accessed 20 June 2020), https://github.com/rukmarr/l3ns

[5] MADT GitHub repository (Last accessed 20 June 2020), https://github.com/dltcspbu/madt

[6] Pumba GitHub repository (Last accessed 20 June 2020), https://github.com/alexei-led/pumba

[7] Fish Recognition Ground-Truth data (Last accessed 20 March 2021), http://groups.inf.ed.ac.uk/f4k/groundtruth/recog

[8] Fish Species Recognition (Last accessed 20 March 2021), http://www.perceivelab.com/datasets

[9] Ozfish (Last accessed 21 March 2021), https://aims.github.io/ozfish

[10] Fish Dataset (Last accessed 22 March 2021), https://wiki.qut.edu.au/display/raq/Fish+Dataset

[11] Kepler.gl (Last accessed 22 March 2021), https://github.com/keplergl/kepler.gl

[12] LP DAAC - SRTMGL1 (Last accessed 22 May 2021), https://lpdaac.usgs.gov/products/srtmgl1v003

[13] Al-Rakhami, M., Gumaei, A., et al.: A lightweight and cost effective edge intelligence architecture based on containerization technology. World Wide Web pp. 1–20 (2019)

[14] Azad, R., Fayjie, A.R., Kauffmann, C., Ben Ayed, I., Pedersoli, M., Dolz, J.: On the texture bias for few-shot cnn segmentation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2674–2683 (2021)

[15] Bakiev, M., Khasanov, K.: Comparison of digital elevation models for determining the area and volume of the water reservoir. International Journal of Geoinformatics 17(1), 37–45 (2021)

[16] Barrachina-Muñoz, S., Wilhelmi, F., Selinis, I., Bellalta, B.: Komondor: a wireless network simulator for next-generation high-density wlans. In: 2019 Wireless Days (WD). pp. 1–8. IEEE (2019)

[17] Benjamin, J., O'Leary, M., McDonald, et al.: Aboriginal artefacts on the continental shelf reveal ancient drowned cultural landscapes in northwest australia. PLoS One 15(7), e0233912 (2020)

[18] Cao, S., Zhao, D., Sun, Y., Liu, X., Ruan, C.: Automatic coarse-to-fine joint detection and segmentation of underwater non-structural live crabs for precise feeding. Computers and Electronics in Agriculture 180, 105905 (2021)

[19] Dong, N., Xing, E.P.: Few-shot semantic segmentation with prototype learning. In: BMVC. vol. 3 (2018)

[20] Ghorbani, M.A., Deo, R.C., Kim, S., Kashani, M.H., Karimi, V., Izadkhah, M.: Development and evaluation of the cascade correlation neural network and the random forest models for river stage and river flow prediction in australia. Soft Computing pp. 1–12 (2020)

[21] Iakushkin, O., Malevanniy, D., Bogdanov, A., Sedova, O.: Adaptation and deployment of panda task management system for a private cloud infrastructure. pp. 438–447. Springer International Publishing (2017)

[22] Islam, M.J., Edge, C., Xiao, Y., Luo, P., Mehtaz, M., Morse, C., Enan, S.S., Sattar, J.: Semantic segmentation of underwater imagery: Dataset and benchmark. arXiv preprint arXiv:2004.01241 (2020)

[23] Islam, M.J., Enan, S.S., Luo, P., Sattar, J.: Underwater image super-resolution using deep residual multipliers. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 900–906. IEEE (2020)

[24] Jian, M., Liu, X., Luo, H., Lu, X., Yu, H., Dong, J.: Underwater image processing and analysis: A review. Signal Processing: Image Communication p. 116088 (2020)

[25] Koganty, R., Alex, N., Su, C.H.: Framework for networking and security services in virtual networks (Feb 12 2019), uS Patent 10,203,972

[26] Lang, D., Jiang, H., Ding, W., Bai, Y.: Research on docker role access control mechanism based on drbac. In: Journal of Physics: Conference Series. vol. 1168, p. 032127. IOP Publishing (2019)

[27] Li, C., Anwar, S., Porikli, F.: Underwater scene prior inspired deep underwater image and video enhancement. Pattern Recognition 98, 107038 (2020)

[28] Li, X., Wei, T., Chen, Y.P., Tai, Y.W., Tang, C.K.: Fss-1000: A 1000-class dataset for few-shot segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2869–2878 (2020)

[29] Lipp, B., Blanchet, B., Bhargavan, K.: A mechanised cryptographic proof of the wireguard virtual private network protocol. In: 2019 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 231–246. IEEE (2019)

[30] Liu, S., Yu, J., Ke, Z., Dai, F., Chen, Y.: Aerial–ground collaborative 3d reconstruction for fast pile volume estimation with unexplored surroundings. International Journal of Advanced Robotic Systems 17(2), 1729881420919948 (2020)

[31] Mahmud, R., Buyya, R.: Modelling and simulation of fog and edge computing environments using ifogsim toolkit. Fog and edge computing: Principles and paradigms pp. 1–35 (2019)

[32] Malevanniy, D., Iakushkin, O., Korkhov, V.: Simulation of distributed applications based on containerization technology. In: Computational Science and Its Applications – ICCSA 2019. pp. 587–595. Springer International Publishing (2019)

[33] Malevanniy, D., Sedova, O., Iakushkin, O.: Controlled remote usage of private shared resources via docker and novnc. In: Computational Science and Its Applications – ICCSA 2019. pp. 782–791. Springer International Publishing (2019)

[34] Miao, J., Wei, Y., Yang, Y.: Memory aggregation networks for efficient interactive video object segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10366–10375 (2020)

[35] Nocerino, E., Menna, F., Chemisky, B., Drap, P.: 3d sequential image mosaicing for underwater navigation and mapping. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 43, 991–998 (2020)

[36] Odun-Ayo, I., Geteloma, V., Eweoya, I., Ahuja, R.: Virtualization, containerization, composition, and orchestration of cloud computing services. In: International Conference on Computational Science and Its Applications. pp. 403–417. Springer (2019)

[37] Roach, T.N., et al.: A field primer for monitoring benthic ecosystems using structure-from-motion photogrammetry. JoVE (Journal of Visualized Experiments) (170), e61815 (2021)

[38] Sedayao, J.C., Smith, C.A., et al.: Virtual core abstraction for cloud computing (Jan 8 2019), uS Patent 10,176,018

[39] Shu, C., Yu, K., Duan, Z., Yang, K.: Feature-metric loss for self-supervised learning of depth and egomotion. In: European Conference on Computer Vision. pp. 572–588. Springer (2020)

[40] Son, J., Buyya, R.: Latency-aware virtualized network function provisioning for distributed edge clouds. Journal of Systems and Software 152, 24–31 (2019)

[41] Urbina-Barreto, I., Chiroleu, F., Pinel, R., Fréchon, L., Mahamadaly, V., Elise, S., Kulbicki, M., Quod, J.P., Dutrieux, E., Garnier, R., et al.: Quantifying the shelter capacity of coral reefs using photogrammetric 3d modeling: From colonies to reefscapes. Ecological Indicators 121, 107151 (2021)

[42] Yang, Z., Wei, Y., Yang, Y.: Collaborative video object segmentation by foreground-background integration. In: European Conference on Computer Vision. pp. 332–348. Springer (2020)