Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
«____» _____ 2024 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 «Информатика и вычислительная техника»

«Система распознавания и решения капч на основе OpenCV»

Руководитель	подпись, дата	доцент, канд. техн. наук должность, ученая степень	Н.Ю. Сиротинина
Выпускник	подпись, дата		А.Н. Головань
Нормоконтролер	подпись, дата	доцент, канд. техн. наук должность, ученая степень	Н.Ю. Сиротинина

Красноярск 2024

РЕФЕРАТ

Выпускная квалификационная работа по теме «Система распознавания и решения капч на основе OpenCV» содержит 51 страницы текстового документа, 37 иллюстраций; 1 приложение; 2 таблицы, 22 использованных источника.

КАПЧА, OPENCV, РАСПОЗНАВАНИЕ, SOLVER.

Цель работы: разработка специализированного сервиса для решения капч, который будет взаимодействовать с платформой по тестированию.

В первой главе проведен анализ задания, аналитический обзор существующих решений, определены сравнительные характеристики рассмотренных аналогов.

Во второй главе представлен процесс реализации платформы по распознаванию капч, настройки ее компонентов, приведено описание общего алгоритма работы системы.

В третьей главе описана реализация модуля распознавания капч.

В четвертой главе приведены результаты тестирования сервиса.

В системе реализованы функции для решения капч посредством POST-запросов, обеспечивая эффективную обработку и распознавание различных типов капч. При этом особое внимание уделено подробной документации, описывающей принципы работы системы и предоставляющей руководства по её использованию. Система успешно внедрена в эксплуатацию в организации EvoSoft, что подтверждается актом о внедрении.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С появлением большого количества интернет-ресурсов, используемых для коммерческой деятельности, социальных взаимодействий и поиска информации, возникла необходимость в их защите от ботов. Боты — это программы, выполняющие автоматически настроенные задачи, в том числе и мошеннические. Для защиты от таких программ широко используется CAPTCHA [1] (Completely Automated Public Turing test to tell Computers and Humans Apart) – компьютерный тест, предназначенный для различения компьютеров и людей. Этот термин, часто упоминаемый в русскоязычной литературе как "капча", стал общеупотребительным.

С развитием капч возникла и задача автоматизации их решения. Задача решения (распознавания) капчи интересна не только «взломщикам» сайтов и сервисов. В частности, представленная в работе система распознавания капчи реализуется по заказу кампании EvoSoft [2]. Деятельность компании-заказчика реализуемой системы связана с разработкой сайтов и мобильных приложений по доставке еды. Разрабатываемая система не предназначена для несанкционированного доступа к внешним ресурсам, а исключительно для использования в строго законных целях – для автоматизации и улучшения процессов внутреннего тестирования наших собственных приложений. Применение технологий распознавания капч является ключевым элементом в процессе разработки таких решений, позволяя эффективно проверять взаимодействие приложений с различными веб-интерфейсами и улучшать их функциональность. В контексте тестирования автоматизация распознавания капчи обеспечивает более высокую точность и скорость проверок, способствуя повышению качества и безопасности разрабатываемых продуктов.

Разнообразие типов капч, от текстовых до сложных графических пазлов, требует разработки многофункциональных и адаптивных методов распознавания. В этом контексте алгоритмы компьютерного зрения, такие как те, что предлагает OpenCV [3], играют ключевую роль, позволяя создавать

системы, способные эффективно интерпретировать и решать различные типы капч.

Целью данной работы является разработка специализированного сервиса для решения капч, который будет взаимодействовать с платформой по тестированию.

В ходе достижения поставленной цели должны быть решены следующие задачи:

1. Изучение предметной области, обзор аналогов, анализ методов решения капч.
2. Проектирование и реализация системы распознавания капч с использованием библиотеки OpenCV.
3. Интеграция с платформой по тестированию.
4. Тестирование разработанной системы, исследование возможностей адаптации к различным видам и сложности капч.

В системе реализованы функции для решения капч посредством POST-запросов, обеспечивая эффективную обработку и распознавание различных типов капч. При этом особое внимание уделено подробной документации, описывающей принципы работы системы и предоставляющей руководства по её использованию. Система успешно внедрена в эксплуатацию в организации EvoSoft, что подтверждается актом о внедрении (Приложение А).

Планируется дальнейшее развитие и расширение системы, а также ведется техническая поддержка существующей версии.

1 Аналитический обзор

1.1 Анализ задания

Введение платформы по решению капч в процесс нагрузочного тестирования веб-сайтов в компании Evosoft представляет собой стратегически важное решение, направленное на повышение эффективности и автономности тестирования. Существующая инфраструктура тестирования, базирующаяся на Selenoid и Selenium [4], демонстрирует высокую производительность и масштабируемость при тестировании веб-интерфейсов. Однако наличие капч в критически важных функциях сайта представляет собой определённое препятствие для полной автоматизации тестирования. Решение капч вручную или с использованием сторонних сервисов не только занимает дополнительное время, но и влечёт за собой дополнительные расходы, а также зависимость от третьих сторон. Использование данной платформы позволяет не только расширить возможности автоматизации тестирования, включая обработку капч, но и значительно сократить время, затрачиваемое на тестирование, уменьшить нагрузку на инфраструктуру тестирования и оптимизировать расходы.

Для лучшего понимания роли и значимости Selenoid в процессе тестирования важно отметить, что Selenoid представляет собой мощный инструмент для запуска изолированных браузерных сессий в Docker контейнерах. Это решение позволяет легко масштабировать тестирование, обеспечивая высокую доступность и эффективное использование ресурсов. Контейнеризация тестов с Selenoid не только упрощает управление тестовыми средами и версиями браузеров, но и значительно сокращает время запуска тестов, обеспечивая более быстрое получение результатов.

Капчи, которые необходимо решить по заданию заключаются в поиске чего-либо на изображении. Результатом должен быть список координат, куда нужно кликнуть, чтобы ее решить. Пример капчи представлен на рисунке 1.

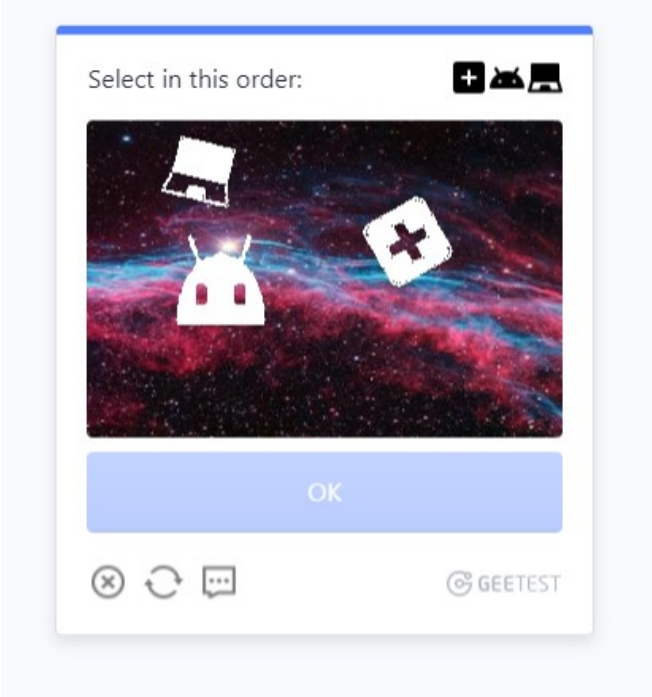

Рисунок 1 – Пример используемых капч

Ключевым моментом в работе системы является обработка капч [5], которая автоматизирована благодаря взаимодействию между контейнерами тестовой платформы и платформой решения капч. При появлении капч в процессе тестирования задача автоматически перенаправляется на платформу решения капч, где она обрабатывается, и результат возвращается обратно в тестовый контейнер для продолжения процесса тестирования. Это позволяет осуществлять тестирование без необходимости вмешательства человека, даже в случаях, когда для доступа к функционалу сайта требуется решение капч.

До внедрения разрабатываемой платформы в процессе тестирования использовалась платформа решения капч стороннего провайдера, который предоставляет свои услуги на платной основе. Использование внешней платформы для решения капч, несмотря на её надёжность, выявило определённые трудности в процессе тестирования, особенно связанные с задержками в работе Selenoid из-за времени, необходимого для обработки капч (в среднем 30 секунд). Это приводило к тому, что тесты оставались в ожидании без выполнения полезных действий, что неэффективно использует ресурсы и увеличивает общее время тестирования. Кроме того, экономическая

составляющая использования платных услуг стороннего провайдера также стала предметом для пересмотра, поскольку внедрение собственной системы решения капч позволит существенно сократить расходы.

Таким образом, интеграция своей платформы для решения капч в систему нагрузочного тестирования является значительным шагом вперёд в развитии тестирования в компании evosoft. Это не только улучшает качество тестирования за счёт полной автоматизации процесса, но и открывает новые возможности для оптимизации и масштабирования тестовой инфраструктуры.

При разработке собственной платформы для решения капч [6] в компании EvoSoft в первую очередь было поставлено условие минимизации человеческого участия в процессе распознавания.

Основной критерий выбора метода заключался в способности обеспечивать высокую скорость обработки при сохранении точности распознавания, чтобы максимально автоматизировать процесс тестирования и сделать его более эффективным

OpenCV позволяет достичь высокой эффективности без необходимости добавления обширных внешних зависимостей и сложных компонентов. Для работы с OpenCV достаточно интеграции небольшого набора библиотек для Python [7], что значительно упрощает развертывание и поддержку платформы.

Задание разработать платформу, способную обрабатывать входящие POST-запросы, содержащие изображения капч, закодированные в формате, с дополнительной информацией о типе капчи. Основная функция системы - проведение детального анализа полученных данных с целью предоставления точных и оперативных результатов для решения капч. При проектировании необходимо учесть возможность масштабирования системы и интеграции новых алгоритмов и методов распознавания капч, обеспечивая тем самым гибкость и адаптивность платформы к развивающимся требованиям и условиям эксплуатации. Дополнительно, платформа должна быть оснащена функционалом для сбора и анализа статистики по решенным капчам.

Система должна выполнять следующие задачи:

– обеспечивать сбор и обработку данных капч в реальном времени и представлять результаты в удобной для тестировщика форме;

– необходимо реализовать 2 пути API:

а) /backend/captcha/createTask – этот маршрут предназначен для отправки капчи на сервер для её распознавания;

б) /backend/captcha/result – этот маршрут API используется для опроса статуса решения капчи.

Требования к данным:

– данные должны обрабатываться с сохранением их целостности и точности;

– капча должна быть представлена в виде строки, закодированной в формате base64.

Функциональные возможности системы распознавания и решения капч на основе OpenCV:

– логирование и анализ данных для повышения эффективности распознавания;

– предоставление точных координат или действий для решения капч;

– обработка изображений капч, представленных в виде строки, закодированных в формате base64;

– распознавание и решение различных типов капч.

Нефункциональные требования:

– система должна обрабатывать один запрос в течение 1 секунды;

– точность совпадения координат решения капчи должна иметь погрешность не более 3%.

Экономическая и техническая эффективность:

– разработка должна ориентироваться на создание экономически выгодного решения, стоимость которого будет ниже, чем у существующих аналогов;

– должна быть разработана понятная инструкция или документация по решению капч через систему.

Ожидаемыми результатами внедрения системы распознавания и решения капч являются:

– автоматизация процесса распознавания и решения капч;

– улучшение точности и эффективности в распознавании различных типов капч;

– расширение функциональных возможностей в области обработки капч и адаптация к новым типам.

На первом этапе разработки требуется реализовать распознавание капч следующих типов: Slider, Select, Rotate. Предусмотреть возможность расширения списка.

1.2 Обзор предметной области

Применение капчи широко распространено в интернете для защиты веб-сайтов от спама, автоматических атак и других форм злоупотреблений. Основная задача капчи – создать задачу или тест, который легко решается человеком, но сложен для автоматических систем.

Существует несколько типов капч:

– текстовая капча: пользователю предлагается ввести текст, который отображается на изображении. Этот текст может быть искажен или иметь различные фоновые узоры, затрудняющие автоматическое распознавание. Пример изображен на рисунке 2;

– графическая капча: включает в себя задачи на соответствие изображений, выделение определенных объектов на фотографии и другие визуальные тесты. Пример самописной капчи и Geetest [8] изображен на рисунке 3. Пример Google рекапчи изображен на рисунке 4. Отличие заключается только в том, что google рекапча на каждом сайте одинакова везде по принципу прохождения бота. А самописные и Geetest имеют множество

задач по решению. В ряду капч Google можно также выделить капчи от компаний Funcaptcha [9], Hcaptcha [10];

– аудио капча: предназначена для пользователей с ограниченными возможностями зрения, где требуется распознать и ввести слова или числа, произнесенные в аудиозаписи. Пример изображен на рисунке 5;

– логическая капча: заключается в решении простых логических или математических задач. Пример изображен на рисунке 6.

Рисунок 2 – Пример текстовой капчи

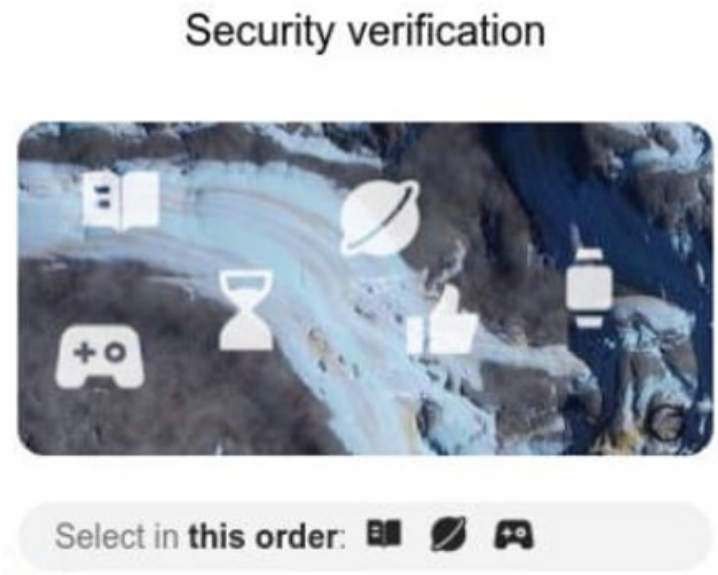

Рисунок 3 – Пример графической самописной или Geetest капчи

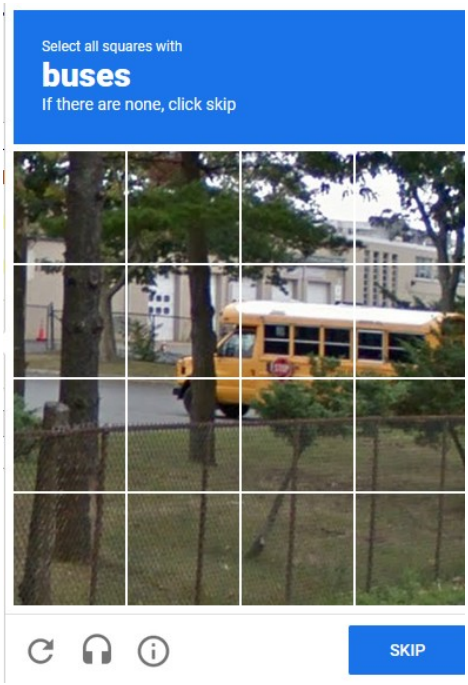

Рисунок 4 – Пример графической Google капчи

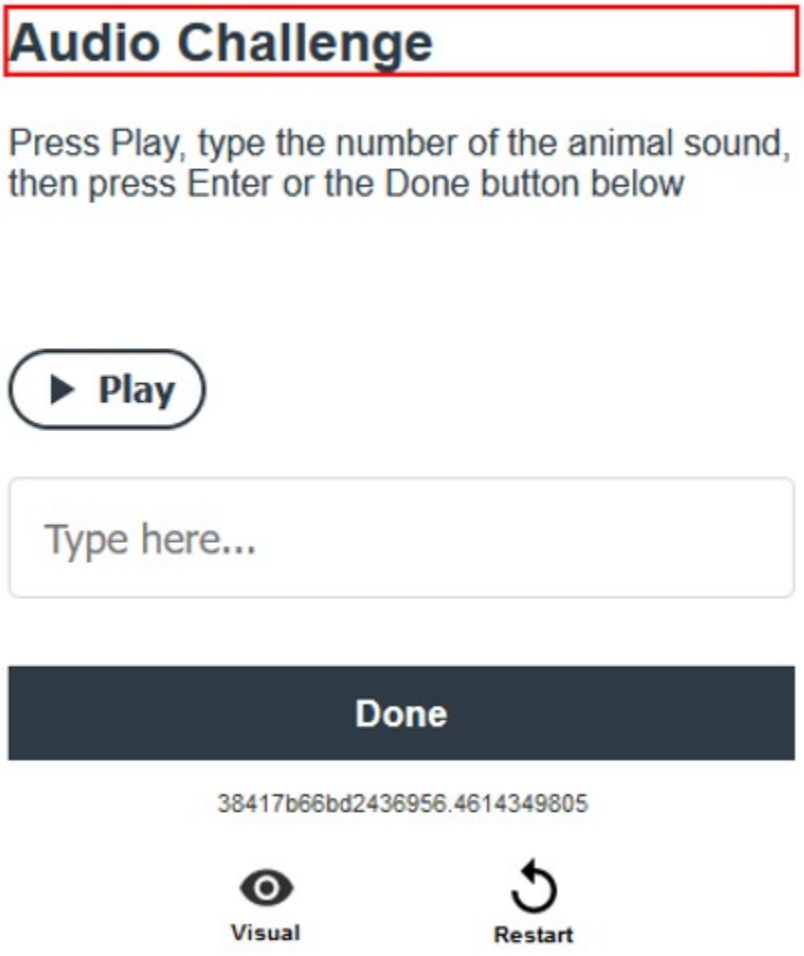

Рисунок 5 – Пример аудио капчи

Рисунок 6 – Пример логической капчи

Ключевыми характеристиками капч являются:

– точность: способность правильно идентифицировать человека и отличить его от бота;

– удобство для пользователя: уровень сложности капчи должен быть сбалансирован так, чтобы не создавать значительных неудобств для реальных пользователей;

– безопасность: способность сопротивляться автоматизированному распознаванию и взлому.

В данной работе затрагивается графический тип капч, а именно самописные капчи и капчи компании Geetest.

Предполагается решение следующих видов капч.

Капча типа "Slider": одна из самых сложных задач – разработка механизма, способного точно определить положение ползунка для совмещения изображений-пазлов. Это требует высокой точности и способности адаптироваться к различным вариациям. Пример капчи типа "Slider" приведен на рисунке 7.

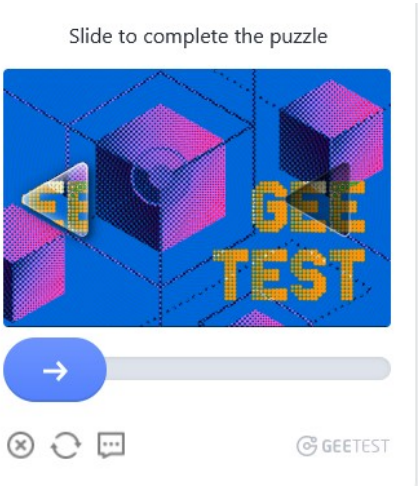

Рисунок 7 – Slider капча

Капча типа "Select": задача заключается в создании алгоритма, который сможет распознавать и правильно выбирать необходимые изображения из предложенного набора. Это требует разработки сложных алгоритмов распознавания, способных работать с различными визуальными элементами. Пример капчи типа "Select" приведен на рисунке 8.

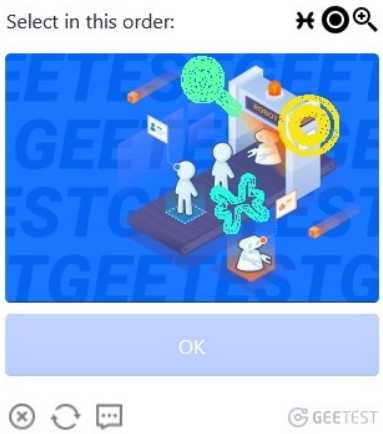

Рисунок 8 – Select капча

Капча типа "Rotate": третья задача - реализация системы, которая может точно определять угол поворота для достижения правильной ориентации изображения. Это требует глубокого понимания геометрии изображений и разработки алгоритмов, способных анализировать и корректировать изображение. Пример капчи приведен на рисунке 9.

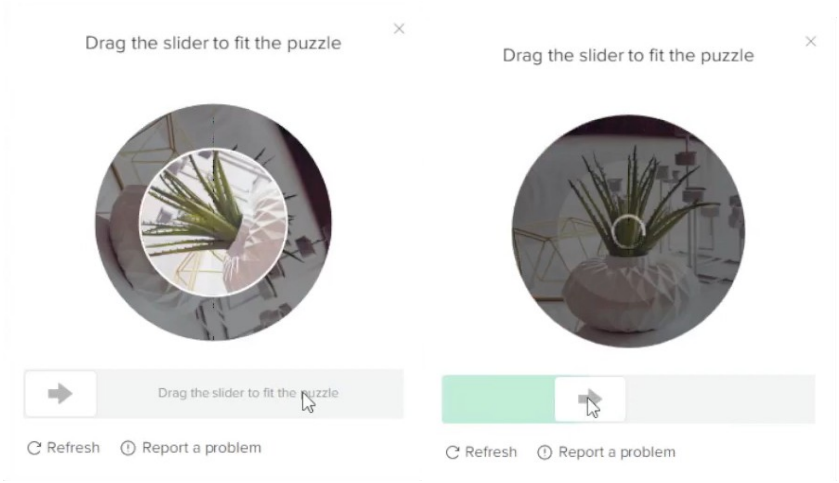

Рисунок 9 – Rotate капча

1.3 Аналитический обзор существующих решений

С целью поиска уже существующих решений, которые, возможно, удовлетворяют требованиям, а в случае отсутствия таковых – для определения возможных методов решения поставленной задачи выполнен аналитический обзор.

1.3.1 Характеристики решений

В оценке и сравнении различных решений для распознавания и решения капч, ключевыми характеристиками являются следующие параметры: скорость, цена, точность и количество поддерживаемых типов капч. Интерес также представляет используемый метод решения капч.

1.3.2 **Сервис Rucaptcha**

Rucaptcha [11]— это сервис, который предлагает решение капч путём привлечения к этому процессу людей. Список и скорость решения поддерживаемых капч изображён на рисунке 10.

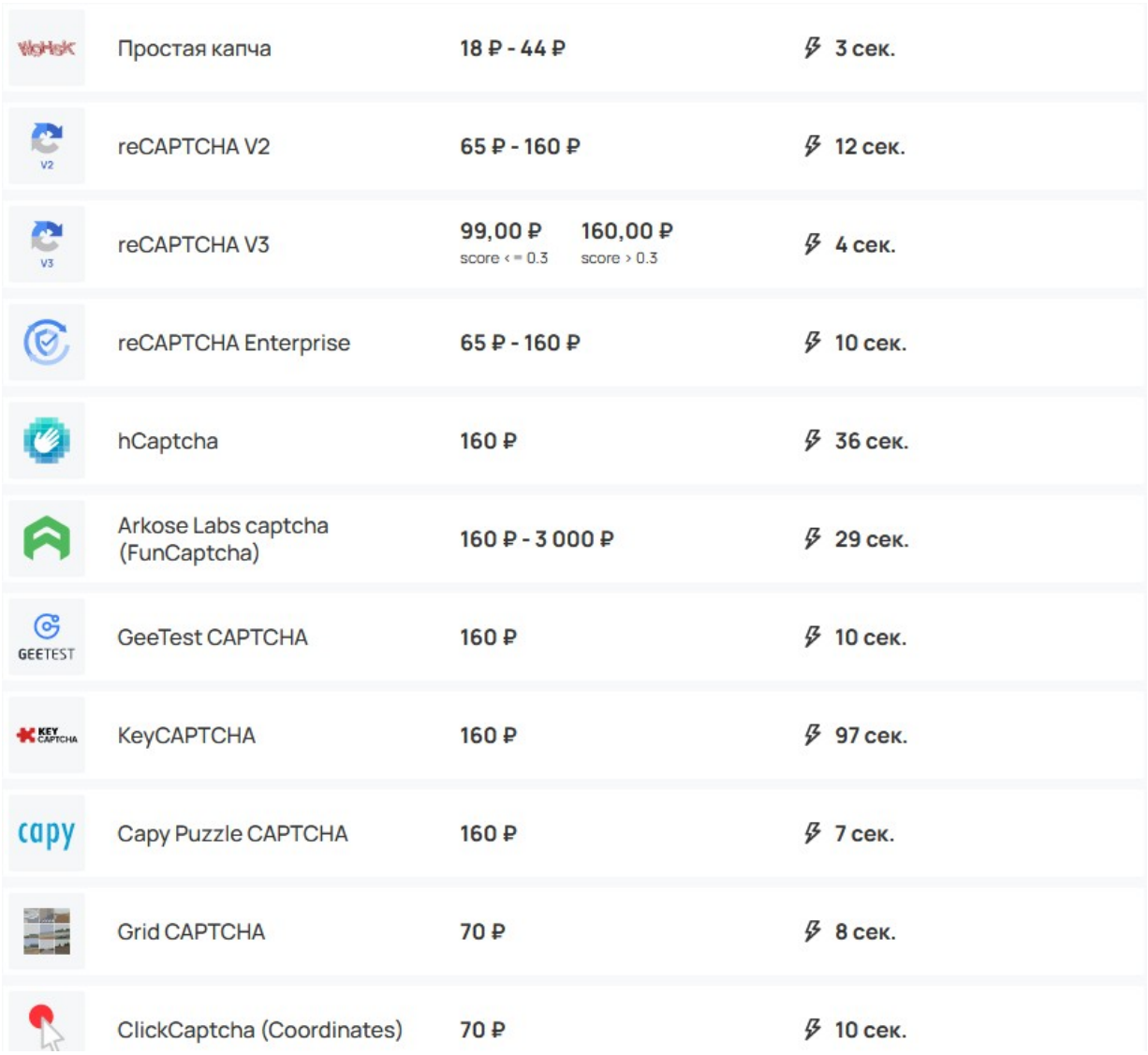

Рисунок 10 – Rucaptcha

Сервис работает по следующему принципу: пользователи отправляют капчи на сервер Rucaptcha, где зарегистрированные на платформе работники, в реальном времени, приступают к их решению. Этот подход обеспечивает высокий уровень точности, так как человеческий фактор позволяет корректно интерпретировать даже самые сложные и искажённые изображения капч, которые могут быть неподвластны автоматизированным алгоритмам.

Следует учитывать, что использование ручного труда обуславливает более высокую стоимость услуг по сравнению с полностью автоматизированными решениями, а также может приводить к увеличению времени ожидания результата. Тем не менее, для многих пользователей и

компаний такие параметры оправдывают себя благодаря почти стопроцентной точности распознавания капч.

1.3.3 Сервис Anticaptcha

Anticaptcha [12] представляет собой сервис, который, так-же как и Rucaptcha, использует ручной труд для решения капч. Список и скорость решения поддерживаемых капч изображён на рисунке 11.

Рисунок 11 – Anticaptcha

Этот сервис предоставляет услуги по распознаванию различных типов капч с помощью операторов-решателей, которые обеспечивают достаточно высокую точность ответов. По сравнению с Rucaptcha, Anticaptcha может предлагать несколько более низкие цены, что делает его привлекательным вариантом для пользователей и компаний, стремящихся сократить расходы при сохранении качественного результата.

1.3.4 Сервис Captchaguru

CaptchaGuru [13] представляет собой современный сервис, цель которого предоставление решений для капч, основанных на использовании технологий искусственного интеллекта (ИИ). Список поддерживаемых капч изображён на рисунке 12.

API	Тип	Поддержка	Цена за 1000
Токены:			
	ReCaptcha v2	Да	30 рублей
	ReCaptcha v3	Да	30 рублей
	hCaptcha	Да	60 рублей
	Turnstile	Да	30 рублей
	Geetest	Нет	30 рублей
Клики:			
	ReCaptcha v2	Да	5 рублей
	hCaptcha	Да	10 рублей
	Geetest	Да	10 рублей
	FunCAPTCHA	Да	8 рублей
	Tiktok	Да	10 рублей
	Binance	Да	10 рублей
	AWS Amazon	Да	5 рублей
	Text img	Да	5 рублей

Рисунок 12 – Captchaguru сайт

В отличие от сервисов, основанных на человеческом труде, CaptchaGuru фокусируется на создании алгоритмов машинного обучения, которые способны самостоятельно анализировать и решать капчи различной степени сложности. Этот подход позволяет значительно сократить время, необходимое для обработки одного запроса, что особенно ценно в условиях высокой нагрузки и когда требуется моментальная обработка больших объёмов данных.

Однако использование искусственного интеллекта может сопровождаться определёнными ограничениями в точности распознавания, особенно когда речь идёт о капчах, специально разработанных для обмана алгоритмов ИИ, таких как те, что содержат сложные визуальные искажения, перекрытия объектов и неоднозначные элементы. В таких случаях, даже несмотря на продвинутые технологии, может потребоваться корректировка человека.

1.4 Сравнительные характеристики рассмотренных аналогов

Сравнительный анализ сервисов Rucaptcha, Anticaptcha и Captchaguru можно оформить в виде сводной таблицы, которая наглядно продемонстрирует ключевые различия и особенности каждого из решений. В таблице 1 представлено описание характеристик различных сервисов распознавания капч.

Таблица 1 – Характеристика сервисов распознавания капч

Параметр	**Rucaptcha**	**Anticaptcha**	**Captchaguru**
Средняя скорость решения в секундах	15	11	3
Процент правильно решенных капч	85	65	70
Количество поддерживаемых видов капч	21	10	5
Средняя цена в рублях за 1000 капч	140	120	40
Используемый метод	Ручной	Ручной	ИИ

Стоимость: этот показатель отражает финансовую затратность использования сервиса для конечного пользователя. Rucaptcha имеет высокую стоимость из-за использования человеческого труда, Anticaptcha предлагает среднюю стоимость, предоставляя более экономичный вариант среди ручных методов, в то время как Captchaguru заявляет о низкой стоимости благодаря автоматизации процесса с помощью ИИ.

Скорость решения: этот критерий оценивает, насколько быстро сервис способен предоставить решение капчи после её получения. И для Rucaptcha, и для Anticaptcha скорость решения является средней из-за времени, которое требуется человеку для интерпретации капчи. Captchaguru обеспечивает более быстрое решение за счёт мгновенного анализа и обработки капч с помощью ИИ.

Точность: одна из самых важных характеристик, указывающая на способность сервиса корректно решать капчи. Rucaptcha гарантирует высокую точность благодаря человеческому взаимодействию, Anticaptcha обеспечивает среднюю точность, в то время как Captchaguru может иметь точность ниже среднего из-за ограничений алгоритмов ИИ перед сложными задачами.

Количество поддерживаемых типов капч: показывает разнообразие капч, с которыми может работать сервис. Rucaptcha поддерживает большое количество различных капч, Anticaptcha имеет средний показатель, а Captchaguru может поддерживать меньшее количество типов, что связано с особенностями обучения ИИ.

Используемый метод: данный параметр описывает основной подход, который используется в сервисе для решения капч. Rucaptcha и Anticaptcha полагаются на ручной труд операторов, в то время как Captchaguru использует методы искусственного интеллекта.

1.5 Вывод

Экономическая выгода от создания собственной системы очевидна: разработка собственного решения не только снижает затраты за счет отказа от платных сторонних сервисов, но и повышает независимость компании. Это особенно актуально в условиях постоянно возрастающего разнообразия капч. Система, способная обрабатывать широкий спектр типов капч, значительно расширяет возможности компании в этой области и позволяет эффективно адаптироваться к меняющимся условиям.

В ходе анализа также были рассмотрены существующие решения для распознавания капч, включая сервисы Rucaptcha, Anticaptcha и CaptchaGuru. Несмотря на их эффективность в определенных сценариях, они имеют ряд ограничений, таких как более высокая стоимость и ограниченные возможности в отношении некоторых типов капч. Это определяет актуальность разработки собственной адаптивной и многофункциональной системы.

Кроме экономических преимуществ, создание собственной системы распознавания капч также улучшает управление данными, повышая уровень безопасности и контроль над обрабатываемой информацией. Это особенно важно в эпоху повышенных требований к защите персональных данных. Вдобавок, наличие внутренней системы позволяет более гибко настраивать алгоритмы под специфические задачи и условия работы компании, что способствует повышению общей производительности рабочих процессов. Таким образом, инвестиции в разработку собственного решения могут окупиться не только за счет снижения операционных расходов, но и за счет улучшения качества тестирования.

2 Реализация платформы по распознаванию капч

2.1 Настройка сервера

Проект использует сервер Nginx [14] на Ubuntu [15] 22.04. Первоначально требовалось настроить передачу HTTP-трафика через TLS. Для этого был получен SSL сертификат через dehydrated и Let’s Encrypt [16] для тестового домена. Сертификат Let’s Encrypt требует настройки Nginx с alias для ACME-challenge. После успешной проверки и регистрации сервера были получены сертификат fullchain.pem и ключ privkey.pem. Затем настроен Nginx для перенаправления трафика с HTTP на HTTPS, завершая базовую настройку сервера.

2.2 Установка зависимостей и настройка окружения

Для работы необходимо установить Python3.10 [17] и все зависимости, указанные в requirements.txt. Получив исходный код проекта, необходимо скопировать пример окружения .env.example в файл .env и настроить следующие переменные окружения:

– API_HOST – домен сервера с API;

– API_PORT – порт сервера с API;

– DJANGO_SECRET_KEY – секретный ключ, используемый Django для шифрования данных;

– PG_DATABASE_NAME – имя базы данных Postgresql;

– PG_USER – имя пользователя Postgresql;

– PG_PASSWORD – пароль пользователя Postgresql;

– PG_HOST – домен сервера с базой данных;

– PG_PORT – порт сервера с базой данных;

– SSL_FULL – путь до SSL ключа + сертификата в одном файле;

– SSL_KEYFILE – путь до ключа сертификата;

– CELERY_WORKERS – количество воркеров для celery.

2.3 Настройка базы данных

Была выбрана реляционная СУБД Postgresql 14 [18]. Для хранения информации о капчах была спроектирована база данных с использованием одной таблицы captchaLog. Эта таблица используется для хранения различных состояний и атрибутов капч, что позволяет эффективно управлять их обработкой.

Таблица captchaLog содержит в себе следующие поля:

– id PrimaryKeyField, первичный ключ;

– captcha CharField, капча, закодированная в формате base64;

– result CharField, результат решения в формате json;

– kind IntegerField, Enum, тип капчи (rotate, slide, select), используется в определении типа капч;

– state IntegerField, Enum, состояние решении капчи (ready, failed, process, done), используется в определении степени решаемости капчи;

– duration IntegerFiled, используется в определении продолжительности решения капчи;

– created_at – DateTimeField, время создания записи в базе данных;

– updated_at – DateTimeField, время последнего обновления записи.

2.4 Настройка асинхронной работы с задачами

В проекте задействован Celery [19] — это инструмент на Python для асинхронной работы с задачами. Когда возникает необходимость решить капчу, она получает в базе данных Postgresql статус "готова к обработке" (state ready), а её идентификатор отправляется в очередь Celery. Как только доходит очередь этой капчи, её статус меняется на "в процессе" (process). Если капча решена, она помечается как "выполнено" (done), а если решить не удалось — как

"неудача" (failed). Также необходимо установить Redis 5 [20], необходим для celery, в нем и происходит распределение очередей для задач.

2.5 Настройка программного интерфейса приложения

Для веб-компонента платформы и управления статистикой использован популярный веб-фреймворк Django 5.0.2 [21]. Этот фреймворк написан на Python и предназначен для создания масштабируемых и удобных для поддержки веб-приложений.

Для интеграции платформы по решении капч в модуль тестирования было реализовано 2 пути API.

/backend/captcha/createTask – этот маршрут API предназначен для отправки капчи на сервер для её распознавания. Контейнеры тестирования используют этот запрос для передачи изображения капчи в формате base64, ее тип и другие параметры. После получения запроса система регистрирует задачу на распознавание и возвращает идентификатор задачи (task ID), который затем используется для получения результата.

/backend/captcha/result – этот маршрут API используется для опроса статуса решения капчи.

На рисунке 13 представлен пример, демонстрирующий как платформа собирает статистику.

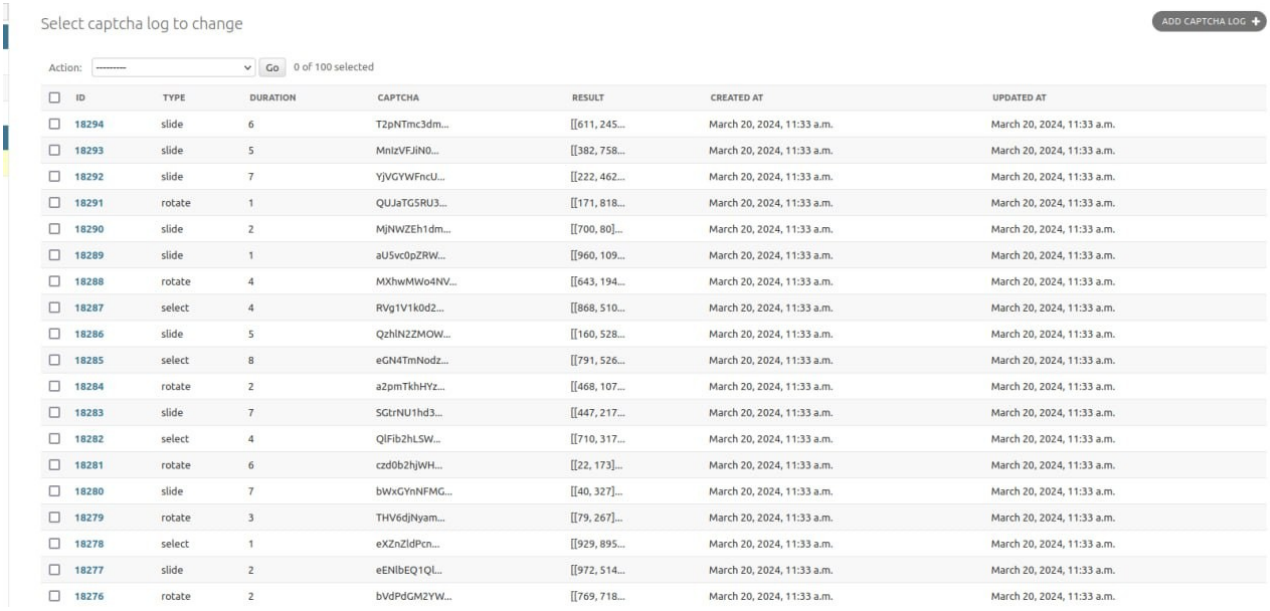

Рисунок 13 – Статистика платформы

2.6 Описание алгоритма работы системы

Модуль тестирования предусматривает использование компонента "Worker spreader", который отвечает за распределение задач по запуску тестов между различными контейнерами на серверах. Этот механизм позволяет оптимально использовать доступные ресурсы и повышать производительность тестирования в целом.

Пример взаимодействия платформы тестирования с сервисом по решению капч представлен на рисунке 14. В Данном фрагменте заменилась платформа по решению капч стороннего провайдера, на нашу платформу с использованием OpenCV.

Контейнер тестирования, когда ловит капчу, отправляет ее на платформу по решению капч. Контейнеры каждые пять секунд, обращаются к API-маршруту /backend/captcha/result, передавая идентификатор задачи, чтобы узнать, была ли капча распознана. Если задача обработана и капча распознана, сервис возвращает результат в виде списка координат. В случае, если обработка ещё не завершена, сервис может вернуть состояние задачи, например,

«обработка», что позволяет контейнерам тестирования понять, что необходимо продолжать ожидание.

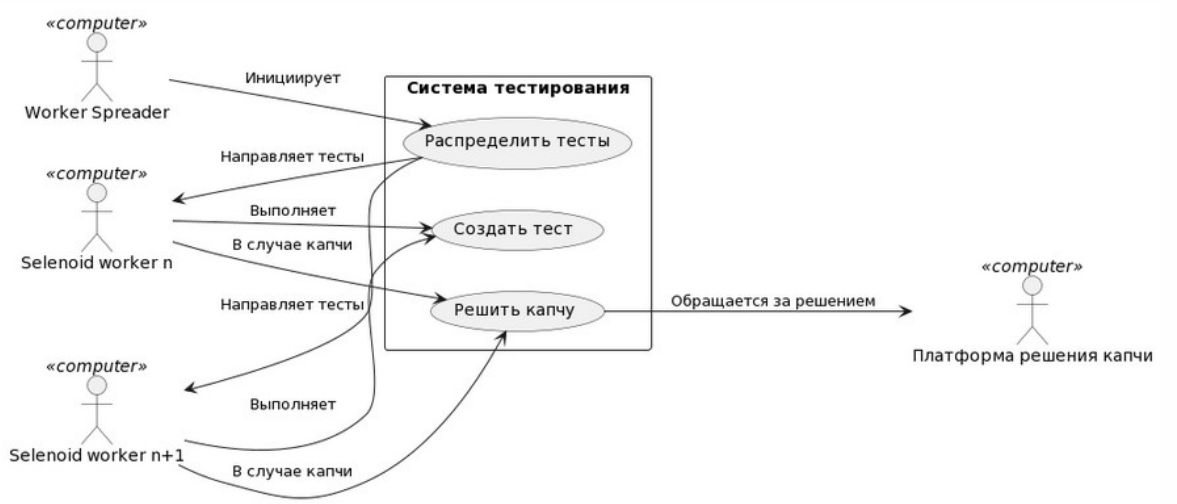

Рисунок 14 – Диаграмма прецедентов платформы тестирования

Общая схема работы контейнера тестирования с платформой по решении капчи представлена на рисунке 15.

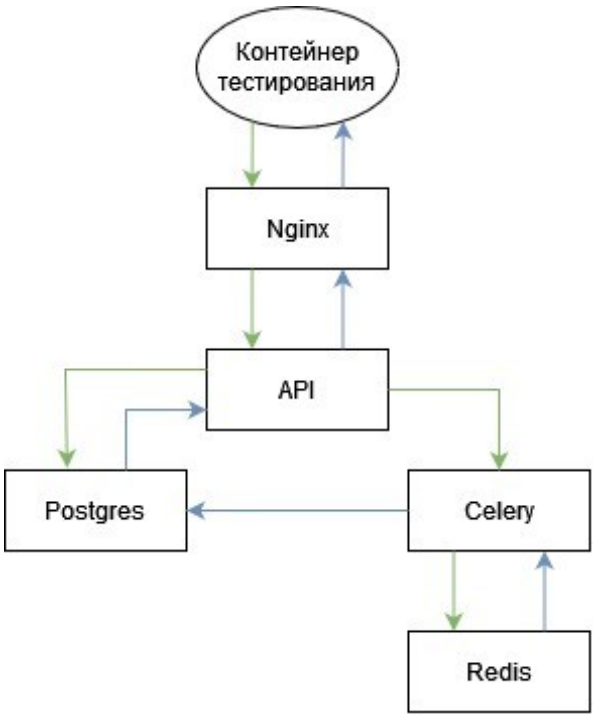

Рисунок 15 – Схема контейнера тестирования

2.7 Вывод

В данном разделе была рассмотрена реализация платформы для распознавания капч, включая настройку сервера Nginx, установку необходимых зависимостей, конфигурацию базы данных PostgreSQL и настройку Celery для асинхронной обработки задач. Ключевые шаги включали получение и настройку SSL-сертификата для обеспечения безопасного HTTPS-соединения, что необходимо для защиты передаваемых данных и соответствия современным стандартам интернет-безопасности.

Была выполнена детальная настройка окружения для работы с Django и Postgresql, что обеспечило надежное хранение и обработку данных капч. Особое внимание уделено настройке Celery с использованием Redis как брокера сообщений, что позволило эффективно распределять задачи между воркерами и управлять параллельной обработкой данных.

API, разработанное на базе Django, предоставляет необходимый интерфейс для взаимодействия с платформой тестирования и управления статистикой, демонстрируя масштабируемость и гибкость используемых технологических решений.

Таким образом, реализация платформы показала свою эффективность и потенциал для дальнейшего расширения функциональности и интеграции с другими системами, что может стать основой для разработки более сложных и многофункциональных систем распознавания капч в будущем.

3 Реализация распознавания капч

3.1 Программные инструменты реализации распознавания капч

Наиболее развитой библиотекой компьютерного зрения в настоящее время является OpenCV.

OpenCV (Open Source Computer Vision Library) – это библиотека программного обеспечения с открытым исходным кодом, предназначенная для компьютерного зрения и машинного обучения. Применение библиотеки OpenCV в данной области представляет собой перспективное направление, поскольку она предоставляет мощные инструменты для обработки изображений и распознавания образов, что может быть использовано для разработки более сложных и надежных систем капч.

Библиотека OpenCV предлагает инструменты для обработки изображений и распознавания образов. Этот метод может включать препроцессинг изображения (например, преобразование в черно-белый формат, удаление шума), сегментацию символов и их классификацию. Преимуществом OpenCV является ее гибкость и доступность, что делает его подходящим выбором для решения стандартных задач распознавания капч.

Прежде всего, OpenCV идеально подходит для работы с капчами, используемыми на наших сайтах. Наши капчи спроектированы таким образом, чтобы быть максимально дружелюбными к пользователю, не требуя сложных и нагруженных решений. Это позволяет избегать использования тяжеловесных сторонних скриптов, сохраняя высокую производительность веб-страниц и независимость от внешних сервисов.

Выбор в пользу библиотеки OpenCV для нашей платформы решения капч был сделан также с учетом стремления к минимализму в интеграции и использовании технологий. В отличие от других подходов, требующих сложной инфраструктуры и больших объемов данных для обучения моделей, OpenCV позволяет достичь высокой эффективности без необходимости

добавления обширных внешних зависимостей и сложных компонентов. Для работы с OpenCV достаточно интеграции небольшого набора библиотек для Python, что значительно упрощает развертывание и поддержку платформы.

Таким образом, выбор OpenCV в качестве основы для платформы решения капч отражает нашу ориентацию на создание эффективной и экономически выгодной системы, которая в полной мере соответствует потребностям компании.

Разработка собственной платформы для решения капч на основе OpenCV обосновывается следующими ключевыми преимуществами.

Высокая скорость обработка капч: технология OpenCV позволяет анализировать и распознавать изображения капч с высокой точностью в реальном времени, что существенно сокращает время, необходимое для прохождения тестов капча должна быть в виде строки формата base64.

Независимость и контроль: разработка собственной системы распознавания капч устраняет необходимость во внешних сервисах, предоставляя полный контроль над процессом и улучшая безопасность данных.

Сокращение очередей и оптимизация ресурсов: автоматизация решения капч снижает нагрузку на серверы Selenoid, позволяя более эффективно распределять ресурсы и уменьшая время ожидания при выполнении тестов.

Экономическая выгода: снижение зависимости от платных сервисов решения капч и оптимизация процесса тестирования ведёт к заметному уменьшению затрат на поддержку и развитие тестовой инфраструктуры.

3.2 Описание реализации

Алгоритм, применяемый для идентификации графических капч, в основном однотипен, при этом ключевое различие заключается в методике обработки цветовой гаммы капчи.

Реализация распознавания графических капч заключается в:

– конвертирование изображение капчи в изображение с оттенками серого;

– применение алгоритма поиска границ изображения;

– поиск наилучшего совпадения заданного шаблона на целевом изображении;

– в случае возникновения трудностей с поиском совпадений, применяются дополнительные методы предварительной обработки изображений, такие как инверсия, масштабирование и корректировка параметров поиска границ, для улучшения распознаваемости капчи.

3.3 Описание алгоритма

Текстовое описание алгоритма включает в себя следующие шаги:

– конвертирование изображения template – изображение, которое ищем; background – изображение, на котором ищем в градации серого;

– обернуть в цикл переменные, которые влияют на решение капчи
Переменные: размер, угол поворота, blur, min_threshold, max_threshold;

– повернуть template на угол поворота I;

– изменить размер на угол поворота j;

– применить блюр на размерность k;

– применить функцию threshold с значениями min_threshold n и max_threshold m;

– применить предыдущие три пункта на изображения template и background;

– применить функцию соответствия match_template, и записать в переменную result;

– найти контуры изображений template и background;

– если количество контуров изображения 0 или больше 15, считать этот результат недействительным (Сделано для того, чтобы не брать результаты, в которых после всех алгоритмов изображение будет пустым, либо нечетким);

– если между блажайшими двумя контурами расстояние больше удвоенного размера template, считать этот результат недействительным

(Сделано для того, чтобы пренебречь изображения, в которых контуры будут расположены по углам);

– если result процент совпадения больше предыдущего лучшего результат, сделать лучшим текущий result;

– конец цикла;

– если лучший результат не определен, state: failed;

– иначе вывести координаты совпадения лучшего результата.

3.4 Пример решения

Для иллюстрации универсальности применяемого алгоритма распознавания капч, рассмотрим процесс решения для капчи типа "Select". Важно подчеркнуть, что хотя предложенный метод является универсальным и подходит для различных типов графических капч, успешное применение алгоритма требует тщательной предобработки каждого конкретного вида изображений капчи. Ключевым аспектом предобработки является адаптация под специфику капчи, что может включать в себя операции такие как переворачивание изображения, корректировка цветовой гаммы, применение фильтров для более четкого выделения границ, а также масштабирование для оптимизации размера исходного изображения.

Для правильного распознавания капчи используются различные функции библиотеки OpenCV. Функции предобработки, в свою очередь, имеют параметры предобработки изображений, каждый из которых играет свою роль в улучшении качества и упрощении последующего распознавания. Правильный подбор этих параметров существенно влияет на результат. Рассмотрим основные параметры и их влияние на результат этапа обработки.

Blur – параметр функции GaussianBlur.

Описание: применение Гауссова размытия для сглаживания изображения. Этот метод использует Гауссово ядро для размытия и уменьшения детализации и шума на изображении.

Зачем нужен: размытие помогает уменьшить шум и неровности на изображении, что может улучшить обработку и распознавание элементов на капче, особенно в условиях низкого качества изображения или при наличии мешающих фоновых текстур.

Возможные значения: размер ядра размытия (обычно нечетные числа, например, 3, 5, 7...). Больший размер ядра приведет к сильнее выраженному размытию.

Thresh – параметр функции threshold.

Описание: пороговое значение для преобразования изображения в черно-белый формат (бинаризация). Пиксели с интенсивностью выше порога становятся белыми, ниже или равными порогу — черными.

Зачем нужен: упрощает изображение, удаляя лишние детали и выделяя важные элементы, такие как текст или объекты на капче.

Возможные значения: от 0 до 255.

Min_threshold – параметр функции canny.

Описание: Минимальное пороговое значение для оператора Canny, используемого для обнаружения краев.

Зачем нужен: определяет минимальную силу градиента, при которой пиксель будет считаться частью края. Помогает исключить слабо выраженные края, уменьшая шум.

Возможные значения: от 0 до максимального порога обычно до 255. (Если threshold тогда 0)

Max_threshold – параметр функции canny.

Описание: максимальное пороговое значение для оператора Canny.

Зачем нужен: определяет порог, выше которого пиксели обязательно считаются краями. Позволяет обнаруживать более четкие и выраженные края.

Возможные значения: от минимального порога до 255. (Если threshold тогда 0)

Angle – параметр переворота.

Описание: угол поворота изображения в градусах.

Зачем нужен: используется для коррекции ориентации изображения капчи, что может быть критично для распознавания текста или объектов.

Возможные значения: от 0 до 360.

Resize – параметр масштабирования.

Описание: коэффициент масштабирования изображения.

Зачем нужен: изменяет размер изображения для оптимизации процесса обработки и распознавания. Может быть необходим для увеличения маленьких деталей или уменьшения размера для ускорения обработки.

Возможные значения: от 0 до n, где значение больше 1 увеличивает размер изображения, меньше 1 — уменьшает.

В совокупности, правильная настройка параметров предобработки изображений играет решающую роль в успешном распознавании капчи. Каждый параметр, будь то пороговая обработка, размытие, обнаружение краев, коррекция ориентации или масштабирование, служит определенной цели и помогает улучшить качество изображения для последующего анализа. Эти параметры вместе работают над упрощением изображения, выделением важных элементов, уменьшением шума и улучшением контрастности, что облегчает распознавание текста, символов или объектов на капче.

Ключ к успешному распознаванию капчи лежит в тонкой настройке этих параметров в соответствии с особенностями конкретного изображения. Неправильный выбор параметров может привести к искажению важных деталей или, наоборот, к недостаточной обработке, что затруднит распознавание и приведет к ошибочному результату. Таким образом, баланс и гармоничное сочетание настроек являются ключевыми для достижения высокой точности распознавания капчи.

Подходящая комбинация параметров обработки позволяет эффективно преобразовать изображение капчи в формат, наиболее подходящий для алгоритмов распознавания, минимизируя вероятность ошибок и увеличивая шансы на успешное решение капчи.

Возвращаясь к задаче, отдел тестирования направляет на платформу данных капчу в формате base64, которая после конвертации представлена в виде изображения, для удобства обозначенного как "Background" (Рисунок 17). Помимо этого, предоставляется задание по поиску определенных элементов (Рисунок 16), именуемых "Templates". Основная цель алгоритма — это определение границ как "Background", так и "Templates" с максимальной точностью.

Анализ представленной капчи позволяет заметить, что элементы для поиска, именуемые "Templates", значительно отличаются по размеру и ориентации от основного изображения "Background". Для достижения максимальной точности в определении границ и успешного распознавания, необходимо провести детальную предобработку данных элементов. Это включает в себя корректировку размеров "Templates" для их приведения к масштабу "Background", а также коррекцию ориентации путем переворота изображений. Такие действия обеспечат более точное сопоставление и идентификацию заданных элементов на изображении капчи.

Рисунок 16 – Templates элементы для поиска на капче

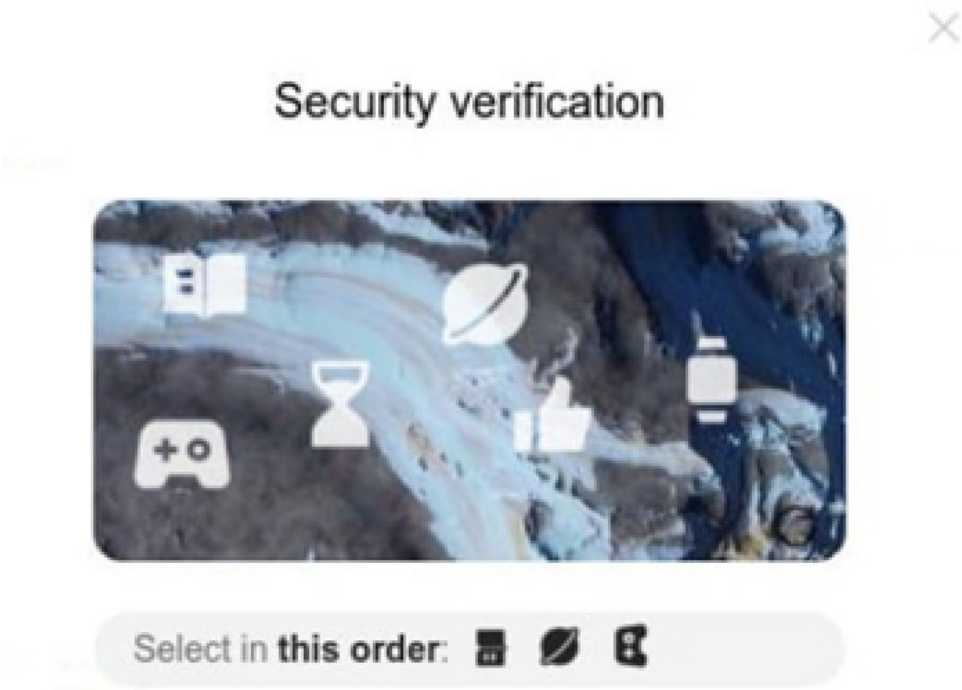

Рисунок 17 – Background изображение, полученное с платформы тестирования

Первым шагом в процессе обработки данных является преобразование "Background" в изображение с оттенками серого (Рисунок 18), что способствует улучшению визуального контраста и облегчает последующий поиск границ.

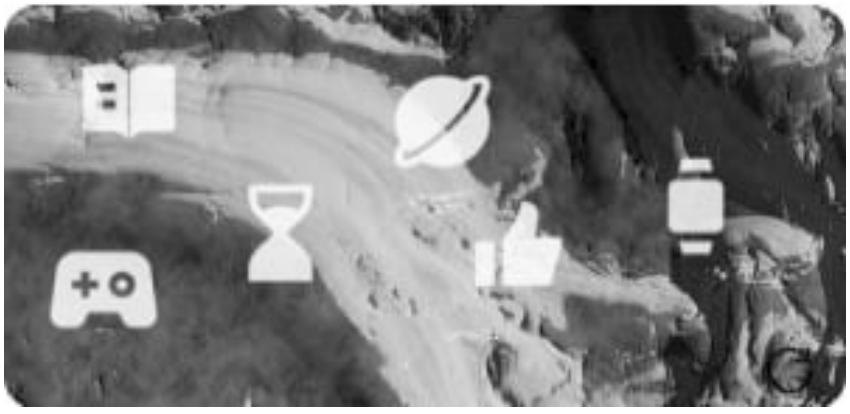

Рисунок 18 – Background в оттенках серого

Продолжая анализ, мы используем алгоритм обнаружения краев Canny, предоставленный OpenCV, что позволяет выделить четкие контуры фона (Рисунок 19). Однако, применение этого алгоритма ведет к появлению избыточной информации, делая распознавание затруднительным.

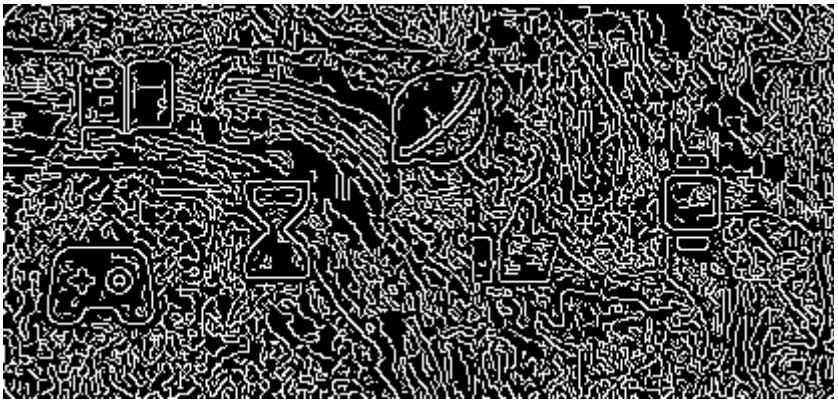

Рисунок 19 – Background после алгоритма поиска границ

Необходимо применить комплексный алгоритм поиска границ, включающий в себя следующие этапы:

– сглаживание изображения для уменьшения влияния шума;

– нахождение градиента изображения для определения направлений наибольшего изменения интенсивности;

– применение немаксимального подавления для исключения слабых границ;

– применение двойного порога для определения сильных и слабых границ.

Начнем с упрощения обработки изображения его преобразуют в черно-белый формат, используя пороговое значение (thresh) 101. Однако, как показано на рисунке 20, даже для человеческого глаза затруднительно различить каждую фигуру корректно, указывая на то, что выбранное пороговое значение не оптимально. Путем экспериментов определяется более подходящее значение параметра thresh, которое делает изображение яснее. При установке порога в 199 становятся видны четкие силуэты фигур, что можно увидеть на рисунке 21.

Рисунок 20 – Background черно-белый с параметром thresh 101

Рисунок 21 – Background черно-белый с параметром thresh 199

Продолжая процесс обработки изображения, следующим шагом является уменьшение уровня шума. Для этого применяем Гауссово размытие с параметром (blur) равным 1. Результат, представленный на рисунке 22, демонстрирует заметное уменьшение шумов на изображении.

Рисунок 22 – Background после применения Гауссова размытия

После снижения шума на изображении переходим к выделению границ. Используя функцию для определения границ, мы получаем изображение с четко очерченными контурами, что видно на рисунке 23.

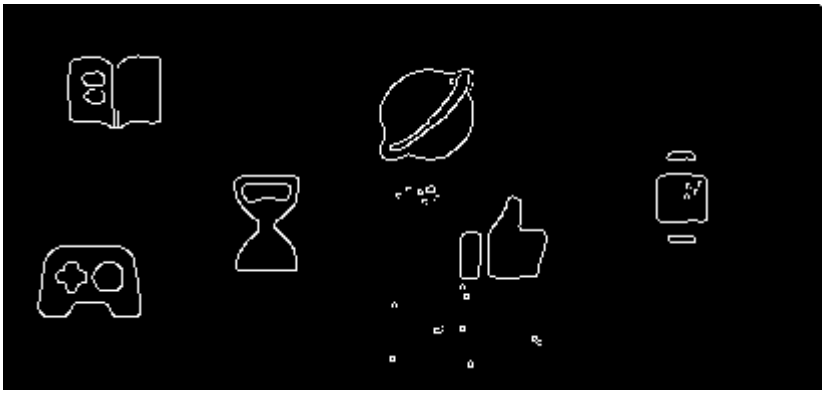

Рисунок 23 – Background с выделенными границами

После того как границы "Background" обработаны, аналогичные действия выполняются для "Templates", что позволяет получить четкое представление об искомых элементах (Рисунок 24).

Рисунок 24 – Templates после алгоритма поиска границ

Заключительный этап алгоритма включает в себя поиск точного соответствия между каждым элементом "Templates" и изображением "Background", а также определение координат обнаруженных совпадений. В этом контексте применяется функционал OpenCV под названием "match_template". Этот метод основан на принципе наложения одного изображения на другое, в результате чего функция выдает координаты точек совпадения и процентное соотношение этого совпадения. Однако важно

отметить, что данная функция не учитывает возможность того, что искомый объект может быть перевернут или изменен в размерах, что требует дополнительной обработки.

Для компенсации этих ограничений, функцию поиска совпадений необходимо встроить во вложенный цикл, который будет поворачивать искомую фигуру на каждые 10 градусов в диапазоне от 0 до 360 градусов, а также масштабировать ее размер на коэффициент 0.1 в диапазоне от 0.5 до 2.5. Это позволит обеспечить более точный поиск соответствий, учитывая потенциальные изменения ориентации и размера искомых элементов на изображении "Background".

Для успешного решения капчи крайне важно подобрать уникальный набор параметров для каждого конкретного случая. Это задача требует тщательного анализа и адаптации алгоритма к условиям и особенностям капчи, что добавляет сложности в процесс распознавания. Ключевыми параметрами являются не только порог бинаризации (thresh) и степень размытия (blur), но и другие операции предобработки, такие как вращение и масштабирование объектов.

Сложность процесса распознавания можно существенно уменьшить, автоматизировав подбор параметров. Это достигается путем внедрения итерационного процесса, где параметры, включая размер и угол поворота, изменяются в заданных пределах с определенным шагом. Такой подход позволяет систематически исследовать пространство возможных настроек и выявлять оптимальные комбинации для каждого случая.

Однако, введение большого количества переменных в цикл обработки повышает сложность задачи, так как требуется не только найти совпадение, но и убедиться, что после всех преобразований изображение остается пригодным для анализа. Это включает в себя проверку на то, что в процессе обработки ключевые контуры не были потеряны или искажены до неузнаваемости. Важно обеспечить, чтобы финальное изображение сохранило в себе достаточно информации для точного распознавания, несмотря на все примененные

трансформации. Это требует дополнительных проверок на наличие и качество выделенных контуров, их размеры и соответствие заданным критериям распознавания.

В результате работы алгоритма формируется массив координат, например: [[42,34], [158,44], [34, 100]], который затем передается обратно на платформу тестирования.

3.5 Значимость правильного подбора параметров

При обработке изображений, особенно при решении капчи, важно понимать, как каждый параметр влияет на целостность и распознаваемость изображения. Приведем примеры изменений изображений в зависимости от различных параметров обработки. Первое изображение в строке – конвертация в черно-белое с порогом вхождения. Второе – алгоритм canny с min_threshold и max_threshold. Третье изображение включает в себя алгоритм первого и второго.

На рисунках 25 и 26 рассмотрим зависимость от blur 0 и 6 значения:

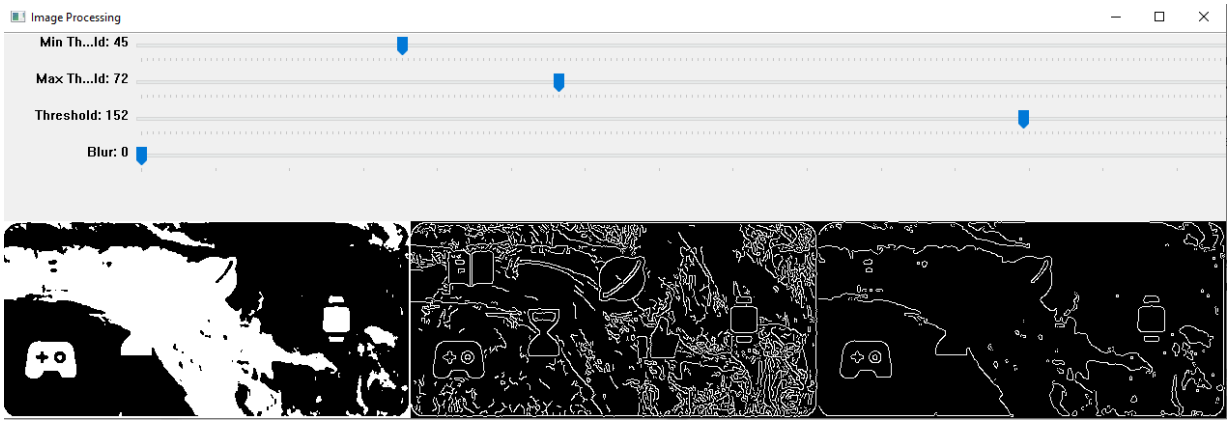

Рисунок 25 – Пример с blur 0

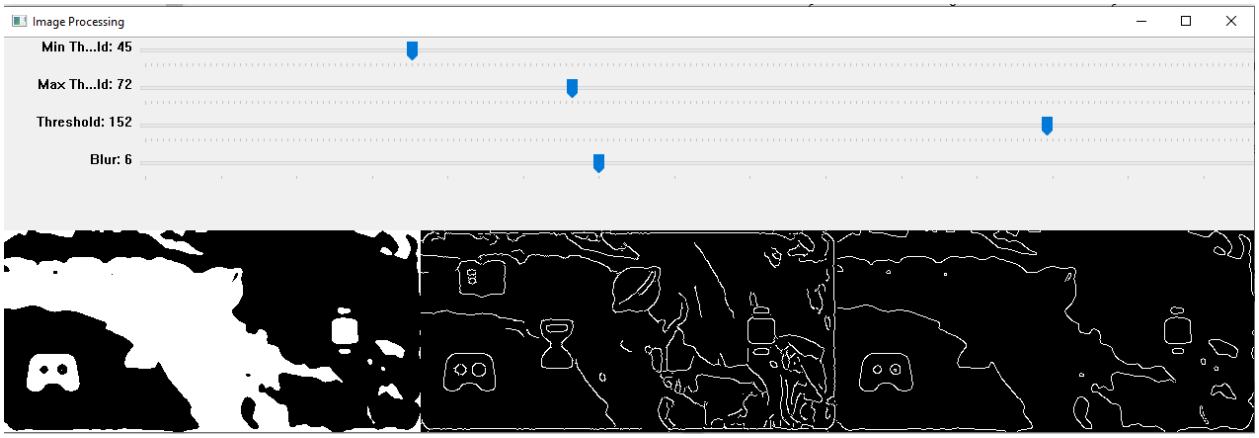

Рисунок 26 – Пример с blur 6

На рисунках 27, 28, 29 рассмотрим зависимость от порогового значения конвертации в черно-белое. Возьмем значения: 90, 170 и 215

Рисунок 27 – Пример конвертации в черно-белое с 90

Рисунок 28 – Пример конвертации в черно-белое с 170

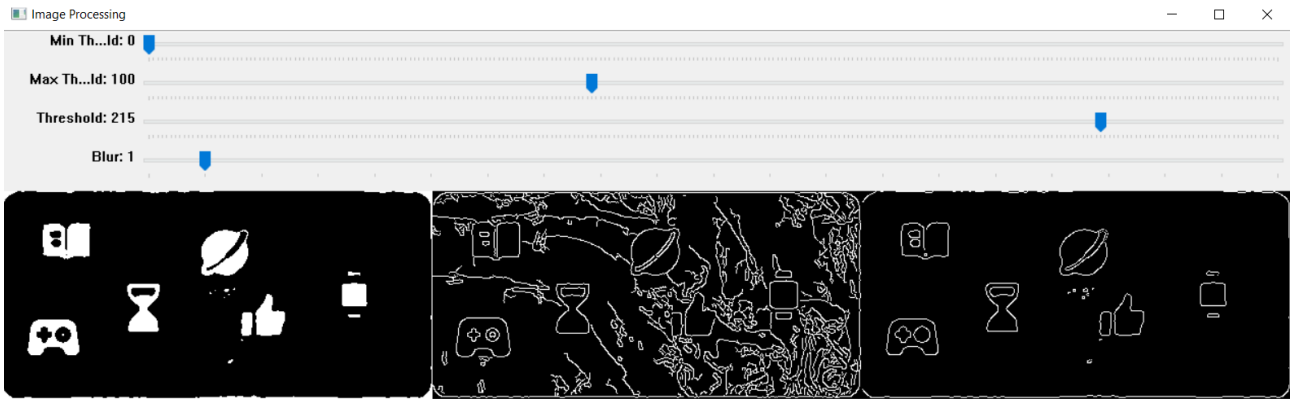

Рисунок 29 – Пример конвертации в черно-белое с 215

На рисунках 30, 31, 32 рассмотрим зависимость от пороговых значений в функции выделения краев canny. Возьмем значения: [10, 110], [10, 200],

Рисунок 30 – Пример выделение краев с [10, 110]

Рисунок 31 – Пример выделение краев с [10, 200]

Рисунок 32 – Пример выделение краев с [80, 110]

Исходя из приведенных скринов увидим, что каждый параметр влияет на то, чтобы сделать капчу нерешаемой. Это необходимо учесть в написании алгоритма по решении.

3.6 Адаптация алгоритма распознавания к разнообразию цветовых схем

В процессе распознавания графических капч важно учитывать, что эффективность алгоритма значительно зависит от специфики цветовой палитры используемых изображений. При этом особенностью обработки является необходимость адаптации к различным цветовым схемам, что требует от алгоритма гибкости и возможности точно выделять необходимые элементы на фоне общей цветовой композиции.

Одним из ключевых аспектов успешного распознавания является способность алгоритма корректно обрабатывать капчи с ограниченным цветовым диапазоном. Так, капчи, выполненные в светлых или темных тонах, обычно проще поддаются анализу. Это связано с тем, что в таких изображениях контраст между символами и фоном выше, что облегчает их сегментацию и последующее распознавание.

В контрасте с этим, капчи, содержащие широкий диапазон цветов и сочетающие в себе как светлые, так и темные элементы, представляют

значительно большую сложность. Разнообразие цветов и оттенков может затруднить выделение отдельных символов, особенно когда цвет символов и фона близок или когда присутствуют различные градиенты и текстуры. В таких случаях требуются более продвинутые методы предварительной обработки изображений, включая адаптивное пороговое значение, фильтрацию шумов и улучшение контрастности, чтобы обеспечить более четкое разделение символов от фона.

Для увеличения точности распознавания в сложных условиях можно применять комбинацию различных техник обработки изображений. Например, использование морфологических операций, таких как эрозия и дилатация, может помочь в устранении шумов и уточнении границ символов.

3.7 Вывод

Описали процесс реализации системы для распознавания капч, используя библиотеку OpenCV, что подчеркивает значительные преимущества такого подхода. Система позволяет эффективно обрабатывать изображения капч, поддерживая высокую точность распознавания в реальном времени. Важной частью реализации является адаптация алгоритма к различным типам капч и настройка параметров обработки изображений, что включает в себя выбор оптимальных значений для бинаризации, фильтрации шумов и обнаружения границ.

Таким образом, разработка собственной платформы для решения капч на основе OpenCV обеспечивает не только технологическую независимость и контроль над процессом распознавания, но и значительное снижение затрат по сравнению с использованием сторонних сервисов. Это делает решение не только эффективным, но и экономически выгодным для организации, учитывая минимизацию внешних зависимостей и оптимизацию ресурсов.

4 Тестирование системы

4.1 Испытание системы распознавания капч

Попробуем с помощью платформы решить select капчу. Капча изображена на рисунке 33.

Рисунок 33 – Пример капчи для решения

С использованием Ruby Faraday воспроизведем запросы на решение капчи. Пример отправки капчи на платформу представлена на рисунке 34.

Рисунок 34 – Пример отправки капчи

После отправки капчи, получаем task_id, поэтому id необходимо запрос результат решения. На следующем примере можно увидеть, что решение капчи находится в state process. Лог изображен на рисунке 35

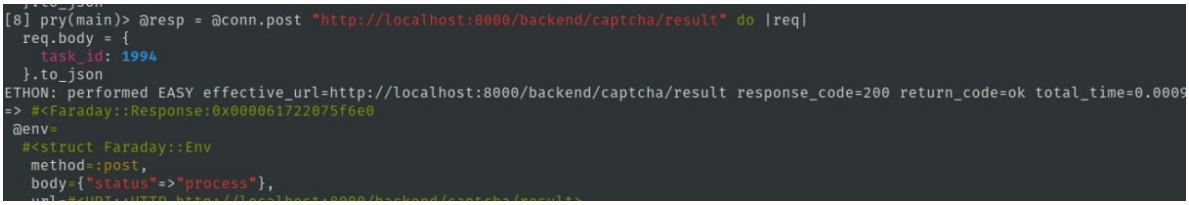

Рисунок 35 – Пример решения капчи в состоянии process

На следующем рисунке можно увидеть готовое решение капчи. Лог изображен на рисунке 36.

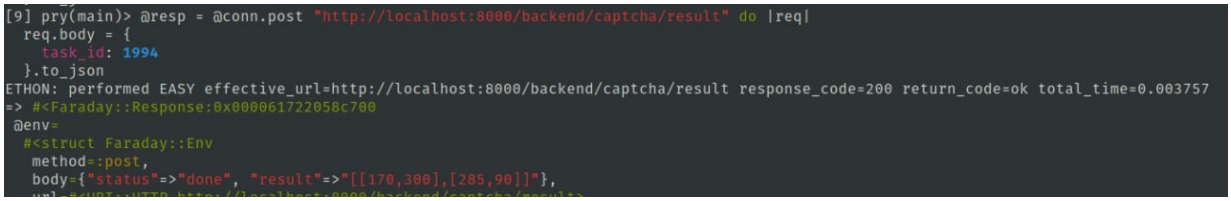

Рисунок 36 – Пример решения капчи в состоянии done

Также может быть вариант, когда капча не сможет решиться, можно увидеть это на рисунке 37.

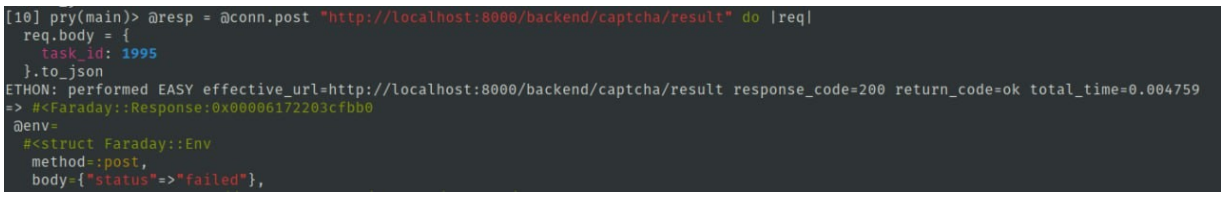

Рисунок 37 – Пример решения капчи в состоянии failed

4.2 Анализ эффективности системы

В рамках анализа эффективности системы распознавания капч было проведено тестирование, включающее 100 тестовых случаев с различными

капчами. Для оценки скорости решения капч была использована разница между временем создания запроса (created_at) и временем его последнего обновления (updated_at). Такой подход позволил точно измерить время, необходимое для обработки каждого запроса. На платформе тестирования также было проведено наблюдение за количеством тестовых случаев, которые завершились неудачно из-за ошибок в распознавании капч. Для проверки погрешности координат были рассмотрены случаи, когда капчи не удалось решить. Ручной запуск процесса решения капч показал, что невозможность подобрать параметры капчи не влияла на точность идентификации объектов, что свидетельствует об отсутствии погрешности в определении координат. В таблице 2 приведены подробные характеристики системы распознавания капч, демонстрирующие её производительность и надежность.

Таблица 2 – Характеристики системы решателя капч

Характеристика	**Система на Opencv**
Средняя скорость решения в секундах	0.5
Процент правильно решенных капч	91
Погрешность в координатах	0

4.3 Вывод

Тестирование системы распознавания капч подтвердило её высокую эффективность и надежность. Использование библиотеки OpenCV и оптимизированных алгоритмов обработки позволило достичь значительной точности в распознавании различных типов капч с минимальным временем ответа. Эти результаты являются основой для дальнейшего развития и совершенствования системы, направленного на улучшение её производительности и адаптации под разнообразные условия применения.

ЗАКЛЮЧЕНИЕ

В заключении выпускной квалификационной работы, посвященной разработке системы распознавания и решения капч на основе OpenCV, можно отметить, что анализ предметной области подчеркивает значительную актуальность и необходимость таких систем. Существующие на рынке решения, в большинстве своем, зависят от человеческого фактора, что приводит к повышенным затратам и задержкам в обработке. Автоматизированная система, разработанная на базе OpenCV, предлагает ряд преимуществ, включая ускорение процесса распознавания капч, снижение стоимости и повышение точности. Это открывает новые возможности для оптимизации различных процессов, где требуется верификация капч, и способствует улучшению общей эффективности систем, использующих такого рода защиту. В целом, разработка такой системы является важным шагом в направлении улучшения и оптимизации процессов автоматизации, связанных с распознаванием и решением капч.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шагалова Е. Н. словарь новейших иностранных слов / Е.Н. Шагалова // AST-Press shkola. – 2018.

2. Evosoft: официальный сайт. – URL: https://evosoft.dev/en/ (дата обращения: 19.12.2023).

3. OpenCV документация : сайт. – URL: https://aerokube.com/selenoid/latest/ (дата обращения: 20.03.2024).

4. Selenoid документация : сайт. – URL: https://aerokube.com/selenoid/latest/ (дата обращения: 20.03.2024).

5. Brodic D., Amelio A. the CAPTCHA: Perspectives and Challenges: Perspectives and Challenges in Artificial Intelligence (Smart Innovation, Systems and Technologies Book) / D. Brodic, A. Amelio // Springer. – 2020.

6. Sandipan D. Python image processing cookbook: over 60 recipes to help you perform complex image processing and computer vision tasks with ease / D. Sandipan // Packt Publishing. – 2020.

7. Joseph H., Joe M. Learning OpenCV 4 Computer Vision with Python 3: Get to grips with tools, techniques, and algorithms for computer vision and machine learning / H. Joseph, M. Joe // Packt Publishing Ltd. – 2020. – 306 с.

8. Geetest capcha : сайт. – URL: https://www.geetest.com/en/ (дата обращения: 23.02.2024).

9. Funcaptcha capcha : сайт. – URL: https://www.arkoselabs.com/ (дата обращения: 23.02.2024).

10. Hcaptcha capcha : сайт. – URL: https://www.hcaptcha.com/ (дата обращения: 23.02.2024).

11. Rucaptcha solver : сайт. – URL: https://rucaptcha.com/ (дата обращения: 19.12.2023).

12. Anticaptcha solver : сайт. – URL: https://anti-captcha.com/. (дата обращения 19.12.2023).

13. Captchaguru solver : сайт. – URL: https://cap.guru/ru/ (дата обращения: 19.12.2023).

14. Reese W. nginx: the high-performance web server and reverse proxy / W. Reese // Linux Journal. – 2008. – Т. 2008. – № 173. – С. 2.

15. Ubuntu : официальный сайт. – URL: https://ubuntu.com/ (дата обращения: 19.12.2023).

16. Letsencrypt сертификаты : сайт. – URL: https://letsencrypt.org/ru/ (дата обращения: 19.12.2023).

17. Python документация : сайт. – URL: https://www.python.org/ (дата обращения: 19.12.2023).

18. Drake J.D., Worsley J.C. practical PostgreSQL / J.D. Drake, J.C. Worsley // Sebastopol, CA: O'Reilly Media, Inc., 2002. – 448 с.

19. Celery документация : сайт. – URL: https://docs.celeryq.dev/en/stable/ (дата обращения: 19.12.2023).

20. Redis документация : сайт. – URL: https://redis.io/docs/ (дата обращения: 19.12.2023).

21. Forcier J., Bissex P., Chun W.J. Python web development with Django / J. Forcier, P. Bissex, W.J. Chun // Boston: Addison-Wesley Professional, 2008. – 416 с.

ПРИЛОЖЕНИЕ А

АКТ О ВНЕДРЕНИИ

ООО «ЭВОСОФТ »

5 мая 2024 г.

ИНН/КПП 2465348570/246501001
ОГРН 1222400019137
https://evosoft.dev
e-mail: alexander.koltashev@gmail.com

АКТ
о внедрении (использовании) результатов выпускной квалификационной работы
Головань Андрея Николаевича
Тема работы: "Система распознавания и решения CAPTCHA на основе OpenCV"
представленную к защите по направлению подготовки
09.03.01 «Информатика и вычислительная техника"

Настоящим актом подтверждаем, что результаты выпускной квалификационной работы «Система распознавания и решения CAPTCHA на основе OpenCV» Головань Андрея Николаевича использованы в деятельности предприятия ООО «ЭВОСОФТ» ИНН 2465348570/246501001 в виде программного обеспечения.

Актуальность разработанного программного обеспечения:

- Большие денежные затраты на решение капчи
- Долгое время тестов
- Отсутствие унифицированного решения

Использование указанных результатов позволило улучшить следующие производственные показатели:

- Точность решения капчи
- Качество предоставляемых услуг

Генеральный директор А. А. Колташев

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
«17» 06 2024 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 «Информатика и вычислительная техника»

«Система распознавания и решения капч на основе OpenCV»

Руководитель _____ 17.06.24 доцент, канд. техн. наук Н.Ю. Сиротинина
подпись, дата должность, ученая степень

Выпускник _____ 17.06.24 А.Н. Головань
подпись, дата

Нормоконтролер _____ 17.06.24 доцент, канд. техн. наук Н.Ю. Сиротинина
подпись, дата должность, ученая степень

Красноярск 2024